

# Detection and Classification of Anomalies in Railway Tracks

**JOSÉ PEDRO DA SILVA MAGALHÃES**

Outubro de 2023



# **Detection and Classification of Anomalies in Railway Tracks**

## **Application of Unsupervised Machine Learning Algorithms in Predictive Maintenance Systems**

**José Pedro da Silva Magalhães**

**Student No.: 1180852**

**Dissertation for obtaining the Master's Degree in  
Artificial Intelligence Engineering**

**Supervisor: Doutora Maria Goreti Carvalho Marreiros, Professora Coordenadora com  
Agregação do Instituto Superior de Engenharia do Instituto Politécnico do Porto**

**Co-Supervisor: Doutor Jorge Alexandre Novo Meira, Professor Adjunto Convidado  
do Instituto Superior de Engenharia do Instituto Politécnico do Porto**

**Evaluation Committee:**

**President:**

Doutora Ana Maria Neves Almeida Baptista Figueiredo, Professora Coordenadora do Instituto Superior de Engenharia do Instituto Politécnico do Porto

**Vocals:**

Doutora Maria Goreti Carvalho Marreiros, Professora Coordenadora com Agregação do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Doutor Diogo Emanuel Pereira Martinho, Investigador do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Porto, October 2023



# Resumo

Em Portugal, existe uma grande afluência dos transportes ferroviários. Acontece que as empresas que providenciam esses serviços por vezes necessitam de efetuar manutenção às vias-férreas/infraestruturas, o que leva à indisponibilização e/ou atraso dos serviços e máquinas, e consequentemente perdas monetárias. Assim sendo, torna-se necessário preparar um plano de manutenção e prever quando será fundamental efetuar manutenções, de forma a minimizar perdas.

Através de um sistema de manutenção preditivo, é possível efetuar a manutenção apenas quando esta é necessária. Este tipo de sistema monitoriza continuamente máquinas e/ou processos, permitindo determinar quando a manutenção deverá existir. Uma das formas de fazer esta análise é treinar algoritmos de *machine learning* com uma grande quantidade de dados provenientes das máquinas e/ou processos.

Nesta dissertação, o objetivo é contribuir para o desenvolvimento de um sistema de manutenção preditivo nas vias-férreas. O contributo específico será detetar e classificar anomalias. Para tal, recorrem-se a técnicas de *Machine Learning* e *Deep Learning*, mais concretamente algoritmos não supervisionados e semi-supervisionados, pois o conjunto de dados fornecido possui um número reduzido de anomalias.

A escolha dos algoritmos é feita com base naquilo que atualmente é mais utilizado e apresenta melhores resultados. Assim sendo, o primeiro passo da dissertação consistiu em investigar quais as implementações mais comuns para detetar e classificar anomalias em sistemas de manutenção preditivos.

Após a investigação, foram treinados os algoritmos que à primeira vista seriam capazes de se adaptar ao cenário apresentado, procurando encontrar os melhores hiperparâmetros para os mesmos. Chegou-se à conclusão, através da comparação da performance, que o mais enquadrado para abordar o problema da identificação das anomalias seria uma rede neuronal artificial *Autoencoder*. Através dos resultados deste modelo, foi possível definir *thresholds* para efetuar posteriormente a classificação da anomalia.

**Palavras-chave:** Manutenção preditiva, Algoritmos não supervisionados, algoritmos semi-supervisionados



# Abstract

In Portugal, the railway tracks commonly require maintenance, which leads to a stop/delay of the services, and consequently monetary losses and the non-full use of the equipment. With the use of a Predictive Maintenance System, these problems can be minimized, since these systems continuously monitor the machines and/or processes and determine when maintenance is required.

Predictive Maintenance systems can be put together with machine and/or deep learning algorithms since they can be trained with high volumes of historical data and provide diagnosis, detect and classify anomalies, and estimate the lifetime of a machine/process.

This dissertation contributes to developing a predictive maintenance system for railway tracks/infrastructure. The main objectives are to detect and classify anomalies in the railway track. To achieve this, unsupervised and semi-supervised algorithms are tested and tuned to determine the one that best adapts to the presented scenario. The algorithms need to be unsupervised and semi-supervised given the few anomalous labels in the dataset.

**Keywords:** Predictive Maintenance, Unsupervised learning, semi-supervised learning, railway tracks



# Agradecimentos

Agradeço primeiramente meus pais, por sempre me terem apoiado e facilitado o meu percurso académico, sem eles isto não seria possível.

À minha namorada, um obrigado enorme por ter estado ao meu lado durante esta fase e me ter motivado.

Por fim, agradeço aos meus orientadores Professora Doutora Goreti Marreiros e Professor Doutor Jorge Meira, por me terem acompanhado e ajudado durante o desenvolvimento do projeto.





# Index

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
1.1	Context .....	1
1.2	Problem Statement .....	3
1.2.1	Introduction to predictive maintenance systems .....	3
1.2.2	Aims and Objectives .....	4
1.2.3	Implementation Approach .....	5
1.2.4	Contributes.....	5
1.3	Dissertation Structure .....	6
<b>2</b>	<b>State-of-the-Art .....</b>	<b>7</b>
2.1	Technical Background .....	7
2.1.1	Anomaly Detection Techniques .....	7
2.1.2	Anomaly Classification Techniques .....	10
2.1.3	Online Learning .....	11
2.2	Related works research .....	12
2.2.1	Research Methodology and results.....	12
2.2.2	Literature Review .....	14
2.2.3	Conclusions.....	17
<b>3</b>	<b>Implemented System and Methodology .....</b>	<b>19</b>
3.1	System's Architecture .....	19
3.2	Hardware, Libraries, and Tools .....	22
3.2.1	Azure .....	22
3.2.2	MATLAB .....	22
3.2.3	Python and Common Libraries.....	22
3.2.4	TensorFlow and Keras .....	22
3.2.5	Scikit-Learn .....	23
3.3	Dataset .....	23
3.3.1	Dataset description .....	23
3.3.2	Privacy and Ethical Considerations .....	29
<b>4</b>	<b>Experimentation and Validation.....</b>	<b>31</b>
4.1	Data used for training and test.....	31
4.2	Pre-processing data .....	32
4.3	Anomaly Detection Tested Models .....	32
4.3.1	One-Class SVM .....	32
4.3.2	Convolutional Autoencoder .....	33
4.3.3	LSTM Autoencoder .....	39
4.4	Validation Metrics .....	41
4.4.1	Anomaly Detection Algorithms Metrics.....	41
4.4.2	Models performance .....	43

4.5	Anomaly classification Approach.....	49
<b>5</b>	<b>Prototype implementation.....</b>	<b>51</b>
5.1	Service Bus .....	51
5.2	Azure Function .....	52
5.3	Anomaly Detection and classification Modules .....	52
5.4	Final remarks .....	54
<b>6</b>	<b>Conclusion .....</b>	<b>55</b>
6.1	Summary .....	55
6.2	Accomplished objectives.....	56
6.3	Future work and improvements.....	56

# List of Figures

Figure 1 - Hyperplane in one-class support vector machine [12] .....	8
Figure 2 – Elbow method diagram example .....	11
Figure 3 – System connections.....	19
Figure 4 – System’s flow.....	20
Figure 5 – Accelerometers in the train [45] .....	23
Figure 6 - Sensor positioned in the back left .....	24
Figure 7 - Sensor positioned in the back right .....	24
Figure 8 - Sensor positioned in the front left.....	25
Figure 9 - Sensor positioned in the front right.....	25
Figure 10 - Sensor positioned in the middle back .....	25
Figure 11 - Sensor positioned in the middle center .....	26
Figure 12 - Sensor positioned in the middle front .....	26
Figure 13 - Anomaly P1 .....	27
Figure 14 - Anomaly P5 .....	27
Figure 15 - Normal vs anomalous data in the front left sensor .....	28
Figure 16 – Normal vs anomalous data in the back left sensor .....	28
Figure 17 – 1D convolution example .....	34
Figure 18 – MAE vs threshold for an anomaly scenario .....	38
Figure 19 – Confusion Matrix representation [53] .....	42
Figure 20 - Service bus message example.....	51
Figure 21 - Anomaly detection and classification modules response.....	53



# List of Tables

Table 1 – Products, Processes, and Services .....	2
Table 2 – Academic databases .....	13
Table 3 – Search String.....	13
Table 4 – Best hyperparameters for OC-SVM .....	33
Table 5 – Best hyperparameters for convolutional AE with 1 timestep .....	37
Table 6 – Best hyperparameters for convolutional AE with 20 timesteps .....	37
Table 7 – Best hyperparameters for LSTM AE with 1 timestep .....	41
Table 8 - Best hyperparameters for LSTM AE with 20 timesteps .....	41
Table 9 – OC-SVM metrics per scenario.....	44
Table 10 – CNN-AE model with 1 timestep, metrics per scenario .....	45
Table 11 - CNN-AE model with 20 timesteps, metrics per scenario .....	46
Table 12 - LSTM-AE model with 1 timestep, metrics per scenario .....	47
Table 13 - LSTM-AE with 20 timesteps, metrics per scenario.....	48
Table 14 – Overview of the models performance .....	48



# Acronyms and Symbols

## List of Acronyms

<b>AE</b>	Auto-Encoder
<b>AI</b>	Artificial Intelligence
<b>API</b>	Application Programming Interface
<b>CNN</b>	Convolutional Neural Network
<b>CPU</b>	Central Processing Unit
<b>DBSCAN</b>	Density-Based Spatial Clustering of Applications with Noise
<b>FURPS</b>	Functionality, Usability, Reliability, Performance, and Supportability
<b>GECAD</b>	Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development
<b>GPU</b>	Graphics Processing Unit
<b>IF</b>	Isolation Forest
<b>ISEP</b>	<i>Instituto Superior de Engenharia do Porto</i>
<b>LOC</b>	Local Outlier Factor
<b>LSTM</b>	Long Short-Term Memor
<b>MAE</b>	Mean Absolute Error
<b>ML</b>	Machine Learning
<b>OCC</b>	One-Class Classification
<b>OC-AE</b>	One-Class AutoEncoders
<b>OC-SVM</b>	One-Class Support Vector Machine
<b>OPTICS</b>	Ordering points to identify the clustering structure
<b>OC-KNN</b>	One-Class K Nearest Neighbor
<b>PCA</b>	Principal Component Analysis
<b>PdM</b>	Predictive Maintenance



<b>PPS</b>	Products, Processes, and Services
<b>PvM</b>	Preventive Maintenance
<b>REST</b>	Representational state transfer
<b>R2F</b>	Run-to-Failure Maintenance
<b>SOTA</b>	State Of The Art
<b>SSE</b>	Sum Of Square Error
<b>SVM</b>	Support Vector Machine
<b>VCEAD</b>	Variable Cumulative Error Anomaly Detection

# 1 Introduction

## 1.1 Context

The proposed project is developed under the project *Ferrovía 4.0* [1], which aims to address the technological and market challenges that the Portuguese railway sector has encountered.

In its entirety, the objective of the *Ferrovía 4.0* is to test the components, tools, and systems developed both on the railway and the trains, in order to improve the transport service. More specifically, the objectives are as follows:

- Reduce carbon footprint and attract more passengers.
- Optimize maintenance costs on trains and the track.
- Monitor infrastructure and rolling stock data.
- Risk detection and control on time, granting safety conditions and warning services.

This project, led by EFACEC Engenharia e Sistemas, S.A., is made up of a vast consortium, with a huge collaboration on the part of different entities:

- *Plataforma Ferroviária Portuguesa*
- Almadesign
- EveloLeo Technologies
- Inegi
- *Infraestruturas de Portugal*
- Inova +
- ISQ
- *Instituto de Telecomunicações*
- Porto Research Technology & Innovation Center
- *Instituto Superior de Engenharia de Lisboa*
- *Instituto Superior de Engenharia do Porto*
- *Técnico Lisboa*
- Itecons
- *Laboratório Nacional de Engenharia Civil*

- Mind For Metal
- MotaEngil
- Nomad Tech
- Solvic
- *Universidade de Coimbra*
- *Universidade do Minho*
- *Faculdade de Engenharia da Universidade do Porto*

The development of Ferrovia 4.0 is divided into five major parts, distributed among members of the consortium. The macrostructure of the project, characterized by Product, Processes, and Services (PPS), is summarized in the following table:

Table 1 – Products, Processes, and Services

PPS	Title	Description
PPS 1	Sustainability of the railway system	Research, development, and demonstration of sustainable energy and environmental solutions for railway transportation
PPS 2	Maintenance Solutions	Railway/infrastructure maintenance planning to control risks and minimize labor costs
PPS 3	Management and alert platforms for critical safety events	Monitoring and control of security conditions in the use of the infrastructure using sensor systems and automatic alerts
PPS 4	Communications and cybersecurity	Development of pioneering and efficient technologies in the areas of communication and cybersecurity applied to vehicles and railway tracks
PPS 5	Valorization and integrated dissemination of project results	Coordination and management of technical, administrative, and financial components, as well as dissemination of the results obtained

This dissertation was developed under the Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development (GECAD), which is a research unit of the consortium member *Instituto Superior de Engenharia do Porto* (ISEP). The efforts of GECAD, in collaboration with other consortium members, are focused on PPS 2, and, therefore, this PPS will be described in more detail in the next topics.

## 1.2 Problem Statement

As it was previously mentioned, this dissertation will focus on PPS 2, which addresses the problems caused by the unplanned maintenance of rolling stock and railways/infrastructure.

The maintenance of rolling stock and railways/infrastructure affects the transportation service that circulates in the actuation areas, possibly causing delays in the services and consequently monetary losses. Therefore, it is necessary to foresee and plan maintenance operations. Consequently, risks can be prevented in time, and maintenance costs optimized.

### 1.2.1 Introduction to predictive maintenance systems

This section intends to contextualize the reader what Predictive Maintenance Systems are, their usefulness, and the advantages of combining these systems with Artificial Intelligence.

#### 1.2.1.1 Predictive Maintenance systems

Different nomenclature and categories of maintenance management strategies can be found in the literature. For this dissertation, the following three will be considered [2]:

- Run-to-failure (R2F) or corrective maintenance: This happens when equipment fails. Although it is the simplest maintenance strategy, it causes a great impact since it stops all the services and requires the replacement of the equipment.
- Preventive Maintenance (PvM): This happens periodically, as the maintenance has a planned schedule to anticipate failures. Even though it is an effective approach to avoid failures, unnecessary maintenance may occur, leading to unnecessary expenses.
- Predictive Maintenance (PdM): This happens only when maintenance is required. It uses predictive tools to monitor continuously a machine or a process, allowing it to determine when should there be maintenance. Additionally, it enables early failure detection using predictive tools based on historical data (such as machine learning techniques), integrity factors (such as visual aspects, wear, and coloration that differs from the original, among others), statistical inference techniques, and engineering approaches.

PdM has several advantages over other maintenance strategies. With this approach, catastrophic failures are prevented, unexpected faults are reduced, and the mean time between failures is maximized, leading to a reduction of workplace accidents and their severity, and a decrease in the number of repairs and the mean time to repair. Consequently, the useful life of the equipment is extended, which results in earnings increase, fewer maintenance interventions and production costs, and more sustainable activity. According to the literature, a predictive maintenance program that is successfully implemented can reduce maintenance expenses by 25% to 30% on average and provide a 1000% return on investment [3].

#### 1.2.1.2 Data-driven Predictive Maintenance systems

The use of artificial intelligence in PdM brings several benefits. With the use of ML, several algorithms can be trained with a large amount of data and be capable of fulfilling prognostics and prediction of failures. Furthermore, it is even possible to estimate the lifetime of the equipment [3], [4].

On the other hand, with Artificial Intelligence (AI) several problems that PdM arises are obstacles. Firstly, the data comes from different sources (e.g.: sensors) and needs to be accurate, which can be hard since noise can be present. Consequently, datasets can be hard to obtain. Also, the data scientist must be aware of the context from which the data is prevented (operational conditions and production environment). Finally, different algorithms address different PdM problems, so there is a need to search for the one that fits the problem [4].

One critical factor in choosing an adequate algorithm is the dataset. For this work, the dataset has a very low number of anomalies. Consequently, the use of a supervised algorithm is not viable. This is a common problem in PdM maintenance since it can be hard to label certain anomalies, or even expensive [5].

For the previous motives, this dissertation focuses on unsupervised and semi-supervised machine learning algorithms. With unsupervised learning, the algorithms are capable of analyzing and clustering unlabeled data, by discovering hidden patterns or data groupings without labeled data [6]. On the other hand, semi-supervised algorithms use labeled and unlabeled data to assist several supervised and unsupervised learning tasks [7]. For the detection and classification process of anomalies, the author detected with the performed research in the following chapter that the “unsupervised” and “semi-supervised” terms are commonly present in related works.

#### 1.2.2 Aims and Objectives

The main objective of PPS 2 is to provide decision support tools for railway managers using different data sources from railway systems, to obtain new information for decision-making. An innovative aspect of this approach is that the collected data comes from the railway track, and also from the trains. The goal is to use Artificial Intelligence (AI) algorithms, namely rule-based systems, Artificial Neural Networks, Machine Learning (ML), and heuristics to obtain a solution capable of taking preventative measures that facilitate the monitoring and maintenance scheduling.

By achieving the main objective, PPS2 will contribute to the transition from a preventive system to a predictive system, allowing constant monitoring of the railway systems, which will only get maintenance when a degree of deterioration is reached. Therefore, PPS 2 delivers software and hardware capable of collecting and analyzing data related to the condition of the components and its interpretation on a management system.

At the present day, the contribution of GECAD to PPS 2 is the development of a solution capable of detecting anomalies on the railway track by using unsupervised ML algorithms and classifying them. GECAD already has contributions with anomaly detection systems and predictive maintenance systems for railway tracks [8]–[10], but it was given the chance to the author to train and test new approaches

Therefore, the main objectives for this dissertation are:

1. Research anomaly detection techniques for predictive maintenance systems.
2. Detect anomalies on the railway track.
3. Research anomaly severity classification techniques for predictive maintenance systems.
4. Classify anomalies on the railway track.

### **1.2.3 Implementation Approach**

For this dissertation, the plan is to train and compare different machine learning and/or deep learning algorithms to detect and classify anomalies on the railway track. These models shall be tuned to get the best performance out of them. Then, the best model shall take part of the final prototype.

The implementation approach follows a prototype development model, and the aim is that at the end of the dissertation, the final prototype can detect anomalies and classify their severity with efficiency and efficacy. Furthermore, this prototype must simulate a real scenario, being connected to a messaging service.

### **1.2.4 Contributes**

This dissertation has as main contributions:

- Research and resume of the best approaches to detect and classify anomalies on a railway track system.
- Proposal of AI algorithms capable of detecting anomalies on the railway track and classifying the severity.
- Contribution to a national project capable of helping the railway systems in Portugal.
- Contribution to other researchers by providing an innovative system capable of being adapted to other areas.

## **1.3 Dissertation Structure**

Besides the Introduction chapter, this dissertation also has a State-of-the-Art chapter, where the algorithms used to implement the described project are described and related works are analyzed. Furthermore, there is an Implemented System and Methodology chapter, where the architecture, functional, and non-functional requirements are specified, libraries and tools used are described, and the dataset is detailed. Also, an Experimentation and Validation chapter exists, with the validation metrics entailed and experimentations for the implementation provided. The final chapter addresses the conclusions of the project, what was accomplished and what is the future work.

## 2 State-of-the-Art

This chapter entails a description of the algorithms used in this project. Furthermore, research on related works regarding anomaly detection and classification in predictive maintenance systems is made, with a detailed description of how that research was elaborated.

### 2.1 Technical Background

Different algorithms were tested to address the problem described in this dissertation. This topic aims to detail them.

#### 2.1.1 Anomaly Detection Techniques

The anomaly detection in this project consists of detecting outliers in time-series data. More precisely, the algorithms must be capable of performing a binary classification. However, the training process will only use data without anomalies. Consequently, this is a one-class classification (OCC) problem.

Several approaches are used to solve OCC problems, and it is possible to distinguish them into four categories [11]:

- **Density estimation methods** – These methods aim to capture the distribution of the target class. New examples are compared with the estimated distribution and the decision is made based on the similarity.
- **Boundary Methods** – These methods estimate the boundary surrounding the target class examples, assuming that the boundary is capable of completely describing the target class.



- **Reconstruction Methods** – Clustering and data modeling are the foundations of these methods. While presuming that the outliers will not satisfy the properties found in the data, they attempt to capture the structure of the target class.
- **Ensemble methods** – These methods use several base classifiers by combining complementary classifiers and consequently obtaining a greater coverage of the target class.

The following sub-sections describe algorithms that are commonly used to address OCC problems and will be tested in this dissertation. The one that has better performance will be used in the final prototype.

#### 2.1.1.1 One-Class Support Vector Machine (OC-SVM)

The Support Vector Machine (SVM) algorithm was introduced by Vapnik and was mostly used to address binary classification problems. Several adaptations were presented in the meanwhile by researchers (e.g.: least-squares SVM, linear programming SVM, etc.). To tackle the OCC problem, the most appropriate adaptation is the One-Class SVM (OC-SVM) [12].

The one-class SVM technique cannot optimize the margin between two classes similar to how it is done when using the regular SVM since only data from one class is given. Instead, the objective is to create a method that provides a function that takes value 1 in a tiny region (in this case, a nominal region), but value -1 elsewhere (a faulty region). Using a kernel function, OC-SVM maps the data into a feature space  $F$ , which is then separated from the origin with the largest margin, also referred to as the hyperplane (Figure 1)[13].

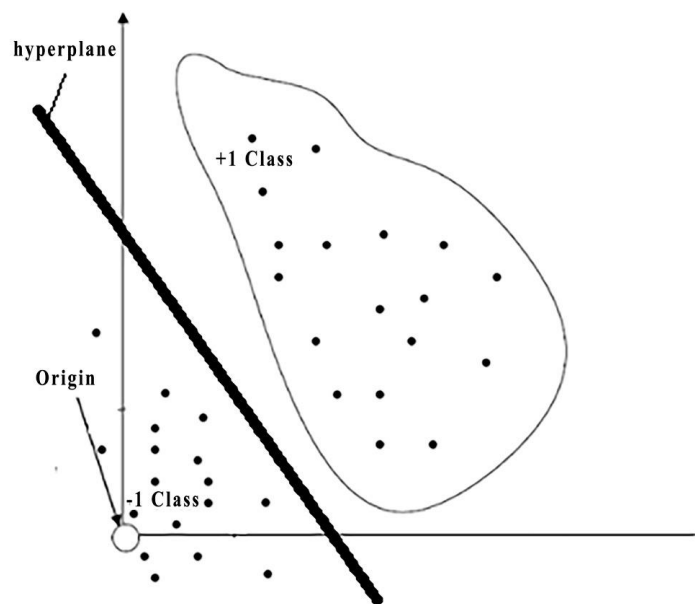


Figure 1 - Hyperplane in one-class support vector machine [12]

#### 2.1.1.2 Local Outlier Factor (LOF)

Local Outlier Factor (LOF) is a density-based method that detects outliers by calculating the local deviation of a given data point. This is appropriate for datasets with unequal distributions to identify outliers. The density between each data point and its neighboring points is used to determine the outlier. The likelihood that a point will be recognized as an outlier increases with a lower point density [14].

#### 2.1.1.3 Isolation Forest (IF)

Isolation Forests (IFs) are tree-based algorithms that can actuate in an unsupervised way. Instead of detecting outliers through basic distances or distance measures, it introduces the concept of isolation. IF generates random isolation trees to isolate each data point. Then, for each tree, the number of branches needed to isolate each of the points is calculated. It is possible to calculate the path length, which is used to isolate a point of interest, by determining the mean of the number of branches. Contrary to anomalous data, the expected path length is generally small for anomalies, since anomalies are far from the majority of anomalous data [15].

#### 2.1.1.4 One-class k-nearest neighbor (OC-KNN)

One-class k-nearest neighbor (OC-KNN) differs from multi-class KNN since its purpose is not to classify different types of data but to determine if the sample belongs to the target data. Therefore, a data boundary for regional division must be established, and if a sample is not within the boundary, then is classified as an anomaly [16].

OC-KNN is provided with a dataset containing only observations of a normal class during training, which is considered the normal class. When a new observation is given to the trained OC-KNN, it will be compared to the training data using a distance metric such as Euclidean distance. If the distance between the new observation and its k-nearest neighbors in the training data is above a pre-defined threshold, the new observation is considered an anomaly [16].

#### 2.1.1.5 Principal Component Analysis (PCA)

Principal Components Analysis (PCA) is a dimension reduction technique that converts a group of initially linked variables into a smaller set of uncorrelated variables called Principal Components. The original variables are combined linearly to create these principal components. Principal components are either fewer or equal in number to the original variables. PCA permits less complexity as a result [17].

To detect anomalies, PCA calculates the statistical distance from each observed data to the normal dataset, that is, the statistical average or centroid. The most common formulas to calculate the distance are Euclidean or Mahanobis distance or statistic  $T^2$  distribution, and the distance calculation is performed in the principal component space. Atypical behavior is deemed to be observed in instances that are far from the new axes represented by principal components. For the comparison, a threshold value is established, and if the calculated distance of the data observed is larger than the threshold, then the data instance is labeled as anomalous.

#### 2.1.1.6 Autoencoder (AE)

Autoencoders (AEs) are a type of neural network trained to reconstruct its inputs and consist of two main parts: an encoder and a decoder. While the encoder maps the input data to a lower dimensional representation, the decoder maps that representation back to the original input space [18].

One-Class AutoEncoders (OC-AEs) are a variation of AEs designed to detect and reconstruct only data that belongs to a specific class or cluster. Since it is trained with only data from one class, it can be used to identify new data points similar to the training data. This algorithm is trained to minimize the reconstruction error of the training data while maximizing the reconstruction error of the anomalous data. With this, the model learns a robust and compact representation of the training data that can be used to detect new data points that are different from the training data. Note that the reconstruction error can be used as an anomaly score [18]. Also, bear in mind that different variations of the AE exist.

### 2.1.2 Anomaly Classification Techniques

One approach to classifying the severity of an anomaly is to compare the anomaly score or distance of the data point in question to a threshold or other data points [19]. The previously described algorithms for anomaly detection provide anomaly scores or distances that can be compared to different thresholds. Different threshold values are equivalent to different severity levels.

On the other hand, anomaly scores can be used as a new label in the dataset, and clustering techniques can be applied. Then, the distance of the data point to the cluster can be used as a measure of severity, or each cluster can correspond to an anomaly state.

The following sub-topics detail different possible clustering approaches.

#### 2.1.2.1 K-means clustering

K-means clustering aims to partition a dataset into  $k$  clusters, where each cluster contains similar data points. The data points are grouped in clusters by comparing their mean value [20].

Some techniques help determine the  $k$ . The elbow method is the most common. This method finds the Sum Of Square Error (SSE) of each data point with its nearest centroid for different values of  $K$ . Increasing the value of  $K$  decreases the SSE and at some point, there is a value for  $K$  where there is the most decline in the SSE, corresponding to the elbow (Figure 2). At this point,  $K$  should not be increased [21].

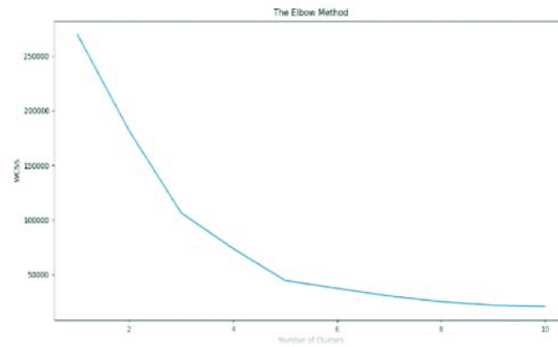


Figure 2 – Elbow method diagram example

#### 2.1.2.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Density Based Spatial Clustering of Applications with Noise (DBSCAN) is the first density-based clustering algorithm. It works by grouping points that are close to each other based on a distance metric. As input, this clustering method requires two parameters: a radius (eps) and the minimum number of points required to form a dense region (minPts). Consequently, points in a dense region are defined as having at least minPts within a distance of eps [22].

#### 2.1.3 Online Learning

When training a machine learning model, the traditional machine learning paradigm frequently operates in a batch learning way. Such a paradigm requires the whole training data set to be made available before the learning task. Additionally, training is frequently carried out offline due to the high cost of training. Traditional learning usually provides as drawbacks low efficiency in both space and time costs and poor scalability for new applications since the model requires to be retrained from scratch to consider new training data [23].

Online learning is a technique of machine learning for data arriving in sequential order, as opposed to batch learning methods, where a learner seeks to learn and update the best predictor for future data at every step. Because the predictive model may be rapidly updated for any new data instances, online learning can overcome the limitations of batch learning. In real-world data analytics applications where data are not only enormous in quantity but also arriving at a high velocity, online learning algorithms are therefore much more effective and scalable for large-scale machine learning tasks [23].

Although not implemented in this project, it is important to consider this approach in the future, since it is beneficial to a predictive maintenance context.

## **2.2 Related works research**

This section addresses the research conducted to find related works regarding predictive maintenance on railway tracks/infrastructures. First, the research methodology will be described.

### **2.2.1 Research Methodology and results**

The research for SOTA is one of the most important parts of this dissertation since it will guide the author on which technologies, approaches, and methodologies are the most commonly used. Therefore, the research process must be carefully thought out first to collect valid information that helps to achieve the best results.

#### **2.2.1.1 Research Aim**

The first thing to bear in mind is what is the focus of this research, and its purpose. More precisely, it is important to define which question(s) should be answered. Within this investigation, the author questions himself “What are the best unsupervised learning approaches to detect and classify anomalies in a predictive maintenance system”. To answer this question, the different domains/topics regarding the question will be investigated:

1. One-class classification problems
2. Unsupervised/semi-supervised anomaly/outlier detection techniques in Predictive maintenance systems
3. Unsupervised/semi-supervised anomaly severity classification in Predictive Maintenance systems

#### **2.2.1.2 Exclusion and Inclusion Criteria**

Now that the main research topics are defined, it is important to decide on exclusion and inclusion criteria regarding works found during the investigation. This step is also of extreme importance since it will help the author exclude/include content that does not contribute to this dissertation more easily and thus make the research process more efficient.

Primarily, the works found must be of relevant topics, which were previously enumerated. Furthermore, it is of extreme importance that the methodologies or results used are described in the papers. Moreover, the sources for the investigation must be in English, since it will rise the probability of finding more works, with more citations, and international interaction. Finally, the date of publication of the work will also be considered. Since this project is intrinsically linked with AI, which is an area of constant growth, progress, and innovative solutions, past works quickly become outdated. Therefore, the author will not consider works conducted before the year 2018.

#### 2.2.1.3 Search Engines

For this investigation, the following academic databases displayed in the presented table shall be used:

Table 2 – Academic databases

Search System	Uniform Resource Locator
Science Direct	<a href="https://www.sciencedirect.com/">https://www.sciencedirect.com/</a>
IEEE Xplore	<a href="https://ieeexplore.ieee.org/Xplore/home.jsp">https://ieeexplore.ieee.org/Xplore/home.jsp</a>

Furthermore, Google Scholar, which is not categorized as a bibliographic database, but as a crawler-based web search engine [24], will also support the research.

#### 2.2.1.4 Search String

By combining various search terms with the Boolean operators OR and AND, the search string used to discover publications that might be included in this systematic review was created (Table 3). Terms related to railway tracks/systems were omitted since if a relevant work on that area exists, then the author will eventually find it with this query.

Table 3 – Search String

Field of Study	Search Terms
Maintenance Strategy	("predictive maintenance")
	AND
Anomaly detection and classification	("anomaly detection" OR "anomaly classification" OR "anomaly severity" OR "fault detection")
	AND
Machine Learning	("Unsupervised Learning" OR "semi-supervised learning")

#### 2.2.1.5 Results

After conducting the research methodology and filtering the valuable papers upon examination, the author determined that 12 papers had value for this state-of-the-art.

### 2.2.2 Literature Review

This topic aims to present works that provide solutions to detect and/or classify anomalies in a predictive maintenance system context.

In [25], the authors present an implementation of a clustering algorithm for anomaly detection on health machines. An interesting approach to this work is that the aging of the robot is taken into account, that is, the authors save records of the machine working normally during a period of time, ignoring anomalies. Succeeding the data collection, three independent data sources (robot controller data, energy data, and vibration data) were used to train a One-Class SVM model, which had bad performance since it assumed an anomaly in cases where the machine worked fine, not taking into account the aging process. To fix this, the authors tried to consider the relations between the sources and verified the data was out of sync since it came from different systems. Consequently, the data was filtered and cleaned and interpolation techniques were used to synchronize the data temporally and on the same scale. The final process was to train an unsupervised k-means algorithm in which each centroid corresponds to a movement of the robot. With this approach, the center of the clusters can gradually be updated with new incoming data (making this a datastream algorithm). On the other hand, detected outliers (deviation from a cluster center) help to determine when maintenance is needed.

A clustering approach is also detailed in [26] to monitor the vibration of an exhaust fan, but, in this case, the anomaly is detected and classified within the same algorithm. The first presented step was to apply a Principal Component Analysis (PCA) to reduce the dimension of the data, without losing the majority of the variance. A  $T^2$  statistic is also applied and determines that the anomalies can be detected 41 observations before. However, the authors implement different clustering algorithms (Hierarchical, k-Means, C-Means, and Model-Based) and all perform better. Contrary to the previous work presented, two methods were used to determine the ideal number of clusters, but, in this case, each cluster corresponds to the state of the anomaly. For instance, if the number is two clusters, then one corresponds to the normal state and the other to the anomalous state. In this work, the result was three clusters, and, according to the authors, one cluster can be classified as a normal state, the other as a warning state, and the final as the anomalous state, providing this way the identification and classification of the anomaly in the same algorithm.

Another clustering approach is provided in [27] for anomaly detection in motor bearings. In this case, the model used is a Gaussian Mixture Model. The justification presented for choosing this method is that it can represent a complex distribution of a power spectrum under normal conditions. It uses a Bayesian Information Criterion to calculate the better k and provides a post-process to update the centroids. According to the authors, the model can detect anomalies with success.

Three one-class SVM algorithms are developed in [28] to understand the decay of a marine gas turbine system. The authors compromise to detect which is the level of decay and which equipment is more decayed. To achieve the objective, three different models are trained to:

- Detect the anomaly
- Classify the anomaly
- Detect which equipment is the most anomalous

Although the authors use a dataset mixed with data corresponding to normal conditions and anomalies, they affirm that the detection model also performs well with data only from normal conditions. To determine the degree and direction of the decay, euclidean distance is used.

Another interesting approach to detecting and classifying anomalies is observed in [29]. In this paper, the authors use a One-Class SVM algorithm to detect the anomaly, and the algorithm Ordering points to identify the clustering structure (OPTICS) to classify it. However, according to the information provided, methods using clustering algorithms have a greater performance when applied to a compacter and informative representation of the input space, which is achieved with AEs. Therefore, the first step and main focus of this paper was to provide representation learning by training a knowledge-induced variational encoder with adaptative sampling. This process comprises labeled data from healthy samplings and unlabeled data whose condition is unknown. This implementation incorporates two implicit supervision features, which are a new loss function and a new sampling algorithm. After this step, the authors train the One-Class SVM with the latent representation of the data, using only healthy labeled data, and are capable of detecting the anomalies. On the other hand, a density-based clustering algorithm is applied, without prior knowledge of the type and number of faults, to the latent representation of the data.

To monitor anomalies and explain them in a chemical production line, [30] uses a Convolutional AE to detect anomalies in a dataset with only healthy data. The specific process to achieve this is by comparing the reconstruction with a threshold. Consequently, it is possible to use the loss as a label and train a supervised model. Then, the authors use a Random Forest to select the variables that present more significative information to explain the reconstruction loss behavior. According to the paper, the results are promising but possess some false positives. One downside is that the model requires to be re-trained each time a machine is restarted.

Another proposed work to detect and classify anomalies on maritime machinery, where the anomalies are unknown, is [31], where in the first phase the authors use an unsupervised algorithm that discovers certain anomalies with the leveraging of relevant features, either individually or with data reduction techniques, which leads to detecting outliers through thresholds on feature values, outlier-aware clustering methods, or outlier detection algorithms such as isolation forest and one-class SVMs. In the second phase, a semi-supervised algorithm (One-Class SVM) was trained, initially without known anomalies, but then with the features that showed better separation of normal and anomalous samples in the first phase. To guide the authors, supervised learning evaluation metrics (precision and recall) are applied, whereas newly discovered anomalies correspond to false positives, which lead to a decrease in precision, although a stable high recall indicates that known anomalies are still detected. On the other hand, false positives are manually labeled, re-setting precision and recall. Note that, according



to the authors, to prevent overfitting of the training data and increase the explainability of the algorithm, feature engineering must be validated against domain expertise, and simpler and interpretable features must be given greater consideration. This approach can lead to false positives since the features are derived to explain unforeseen anomalies. To classify the anomaly, a rule-based technique is used, although it is not detailed.

[32] also implements an AE to detect anomalous behavior, in this case, on wind turbines. The model is trained with healthy data coming from more than 55 elements (wind speed, temperature, ...). It follows the normal process for an AE with only normal data instances, which means that the algorithm fails to reconstruct abnormal input data, meaning that a higher reconstruction error is seen on anomalies. Furthermore, the authors also want to know which feature fails, so they present a novel algorithm named ARCANA.

A more complex implementation of an AE is proposed in [33]. To monitor large mechanical devices, the authors provide an unsupervised online monitoring method named Variable Cumulative Error Anomaly Detection (VCEAD). This architecture automatically sets thresholds based on the reconstruction error of the AE and uses a Time-domain Convolutional Network module for time-series prediction. Moreover, the authors present dynamic error calculations and threshold comparisons to determine the possible impacts of a complex and variable operating environment on the computation. The method performs good results detecting the anomalies and has a clear degradation trend. Furthermore, it is an online method and provides a reconstruction error threshold strategy.

Other authors decided to apply more uncommon approaches to detect and/or classify anomalies. [34] tests the algorithms Isolation Forest and Elliptic Envelope to detect anomalies in aluminum productions. For the dataset, the authors monitor and compute multiple features before breakdowns and define anomaly scores and alerts if necessary. It is presented the idea of evaluation per historical time window, for characterizing the criticality of each variable or feature within a window. Furthermore, it was successfully applied forward variable selection based on the evaluation of the Isolation Forest. However, this model needs to be retrained from time to time.

In [35], the authors affirm that data coming from sensors possess uncertainty. Consequently, models must be capable of detecting the next anomaly and also the uncertainty. Therefore, they train a Long Short-Term Memory (LSTM) network and extend the mean-square-error loss function with variance information. The uncertainty can be detected by defining a threshold for the Euclidean distance by a visual examination of the reconstruction error during the training for every sensor, individually. The uncertainty factor is defined based on the maximum allowed reconstruction error to which each sensor can not surpass and still be considered normal. Note that this can only be implemented in a time series.

Another uncommon implementation is described in [36] and also addresses problems in railway tracks, as this dissertation. Here, instead of using unsupervised techniques to detect the anomaly, the authors use an unsupervised heuristic to classify the data. Then, specialists validate the information. The obtained dataset has a great imbalance since only 3.9% of the

data corresponds to anomalous behavior. Therefore, different techniques are applied to mitigate the imbalance, and then a standard SVM is used to make a binary classification.

### **2.2.3 Conclusions**

After the research for related works, the author concluded that not many provide anomaly detection in predictive maintenance systems with one-class approaches since only 12 valuable papers were found. Furthermore, it was hard to find works that classified the anomaly severity, and even harder works that simultaneously detected the anomaly and then classified it.

Overall, the works that address similar problems like the one described in this dissertation tend to use OC-SVM, AEs, or IF for anomaly detection, but the last is the most uncommon. On the other hand, clustering is also common, but after further analysis, it is not adequate for the anomaly detection module of this project. On the other hand, it may provide the best results for anomaly severity classification, if the anomaly scores obtained in the anomaly detection module are used as labels in the dataset for training.



## 3 Implemented System and Methodology

### 3.1 System's Architecture

This architecture is a Proof of Concept. In this architecture, the author considers a service bus that hypothetically would be constantly updated with data from the train sensors. To dispatch the data, an Azure Function shall be used, due to its simplicity to connect to this type of service. Furthermore, Azure Functions allow you to choose different triggers (timer, new data, etc.). The function dispatches new data to a model capable of detecting anomalies. If the model determines that no anomaly is detected, then the system must log this information. On the other hand, if an anomaly is detected, the severity of the anomaly must be determined. Figure 3 details the connections described.



Figure 3 – System connections

Figure 4 entails in greater detail this project's flow.

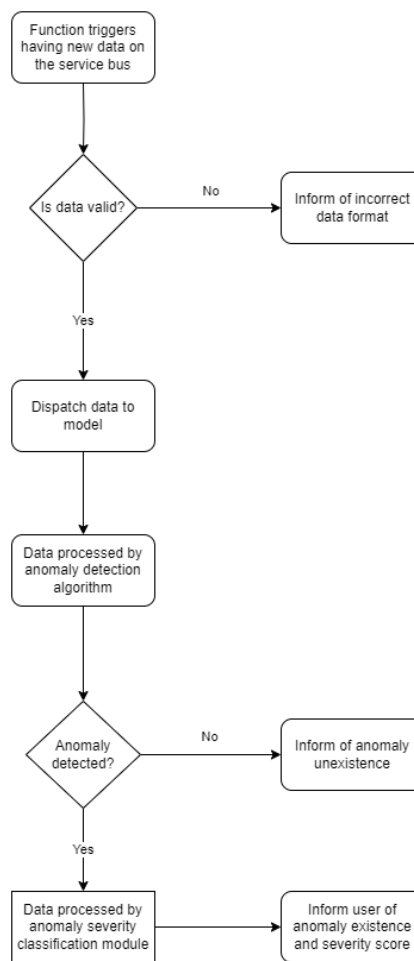


Figure 4 – System's flow

The author decided to use the FURPS+ model to define the project's requirements since it helps do that concisely and clearly, thus facilitating the analysis and later on the design of the solution. FURPS is an acronym for Usability, Reliability, Performance, and Supportability, and the + was added later on and includes three additional categories (Constraints, Interface Requirements, and Business Rules) [37].

#### 3.1.1.1 Functional Requirements

Functional Requirements entail the letter “F” of the FURPS+ model. The ones considered are the following:

- The machine learning/deep learning model must be capable of identifying anomalies based on the input provided by the Azure Function.
- The anomaly classification module must be capable of identifying the anomaly severity based on the input provided by the anomaly detection module.
- The Azure function must be connected to a service bus in Azure, and be capable of communicating with the anomaly detection and classification components.
- All the components must write logs in case of failure.

#### 3.1.1.2 Non-functional Requirements

Non-Functional requirements contemplate the remaining characters (URPS+) of the FURPS+ model.

The Usability Requirements are:

- Authenticated and authorized users must be capable of providing input data to the service bus.

The Reliability Requirements are:

- All the components should be fault-tolerant, restarting automatically if a critical error occurs.

The Performance Requirements are:

- All the algorithms used must be highly efficient.

The Supportability Requirements are:

- All the errors must be logged clearly and concisely.

The Additional Requirements are:

- The anomaly detection and classification module must allow Representational state transfer (REST) Application Programming Interface (API) using HTTP and HTTPS.

## **3.2 Hardware, Libraries, and Tools**

### **3.2.1 Azure**

Azure [38] is a cloud computing platform provided by Microsoft for creating, deploying, and administering applications and services in the cloud, it provides a vast array of services and tools. Azure offers businesses a scalable and adaptable infrastructure on which they can host their programs, store their data, and make use of different computer resources.

Azure Machine Learning Studio is a part of Azure's machine learning offerings. It is a web-based graphical interface that enables users to build, deploy, and manage machine learning models. For this project, Azure Machine Learning Studio was used to train and test the models, since it offers a wide variety of Central Processing Units (CPUs) and Graphics processing Units (GPUs).

### **3.2.2 MATLAB**

MATLAB [39] is a programming and numeric computing platform that allows several scientists and engineers to analyze data, develop algorithms, and create models. The dataset for this project was provided in MATLAB format, so MATLAB was used for the first analysis of the data and to export it to a CSV file.

### **3.2.3 Python and Common Libraries**

The chosen programming language for this project was Python [40], given its versatility and large community. Different Python modules and libraries are required to pre-process the data and provide models to train, test, and deploy.

### **3.2.4 TensorFlow and Keras**

TensorFlow [41] is an open-source machine learning framework developed by Google. It was made to make it easier to create and use machine learning models focusing more on deep learning. The extensive ecosystem of tools, libraries, and resources offered by TensorFlow enables researchers and developers to effectively build and train a wide range of machine learning models.

Keras [42] is a high-level neural networks API that is integrated into TensorFlow. It offers a straightforward interface for creating and refining deep learning models. Initially created as a distinct package, Keras later became TensorFlow's official high-level API.

Note that even for hyperparameter tuning Keras was used. KerasTuner [43] is a hyperparameter tuning library for Keras, which helps in automatically finding the best hyperparameters for a given deep learning or machine learning model. It provides a simple and efficient way to search

and optimize hyperparameters, such as learning rate, batch size, number of layers, and activation functions, using techniques like random search, grid search, and more advanced algorithms like Bayesian optimization. KerasTuner integrates seamlessly with Keras and TensorFlow, making it easier to find optimal hyperparameter configurations and improve model performance.

These were critical tools for training and testing the deep-learning models in this project.

### 3.2.5 Scikit-Learn

Scikit-learn [44] is a Python module, with minimal dependencies, that integrates many state-of-the-art machine-learning algorithms to address supervised and unsupervised problems.

## 3.3 Dataset

This section describes the dataset used in this dissertation, and its privacy and ethical considerations.

### 3.3.1 Dataset description

The used dataset was provided by the Civil Engineering Department of the Faculty of Engineering of the University of Porto. This dataset was developed through mathematical calculations, so it is a simulation and approximation of real occurrences of a train circulating on the railway track.

In more detail, the dataset has data from seven accelerometer sensors in a freight train. The sensors are displayed in the following zones:

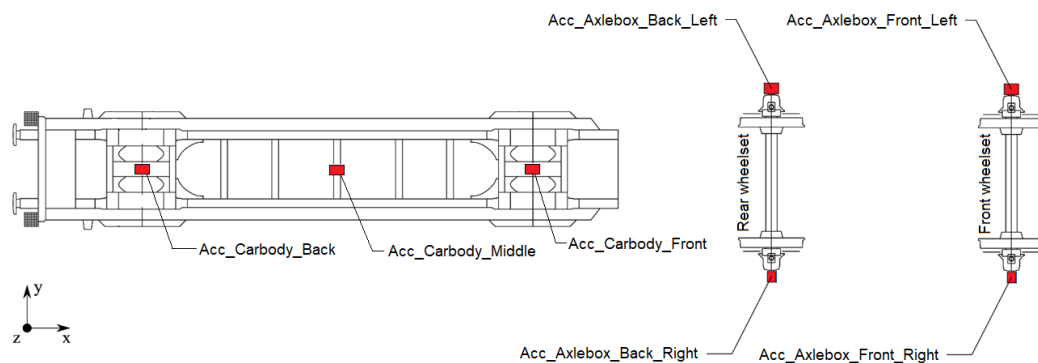


Figure 5 – Accelerometers in the train [45]

From the accelerometer information, it is possible to determine if the railway track has any anomaly. The dataset provides some baseline scenarios for a train with the sensors integrated.



These scenarios are called “Baseline”, and have the following combinations, totaling 200 scenarios:

- Different speeds (km/h): 40, 50, 60, 70, 75, 80, 90, 100, 110, 120.
- Different loads: Empty or full.
- Different irregularities: 10.

Each scenario can be viewed as a dataset itself. The number of records each scenario has depends on the speed of the train since the traveled distance is the same. Furthermore, the number of features is 8 (7 sensors and the timestamp).

The previously mentioned irregularities are not considered anomalies, since they are different types of common singularities that may appear on the track. The following pictures illustrate a different time series for each sensor on the train, for a specific type of irregularity applied on a train going at 75 kilometers per hour:

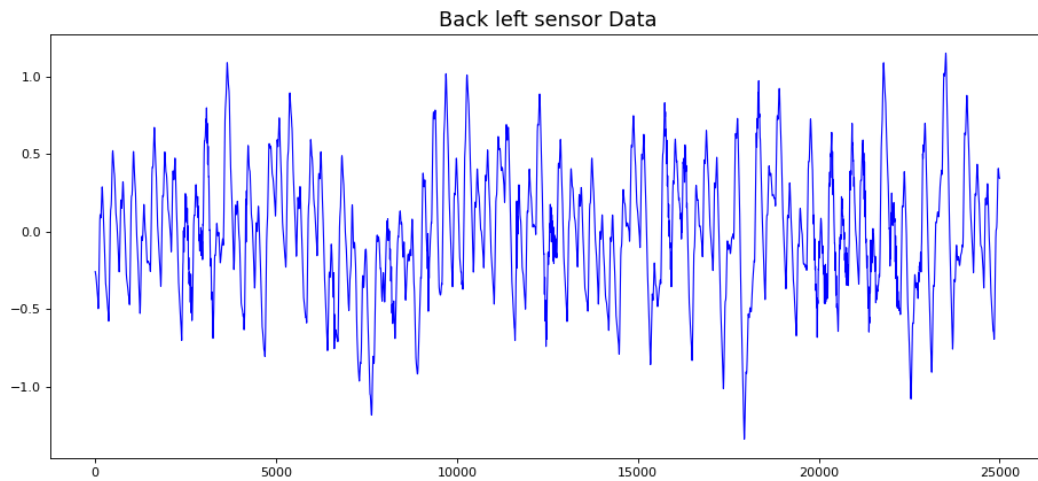


Figure 6 - Sensor positioned in the back left

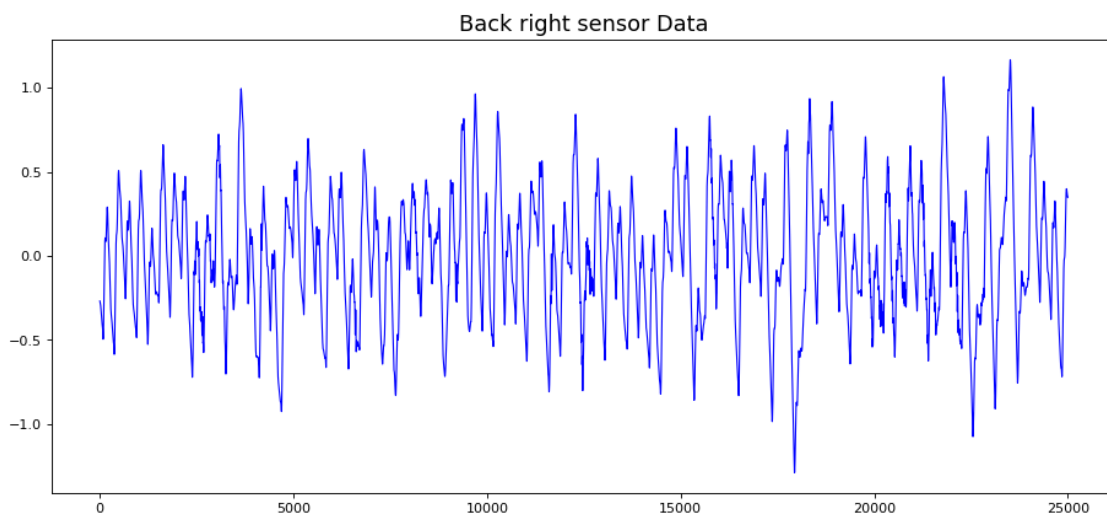


Figure 7 - Sensor positioned in the back right

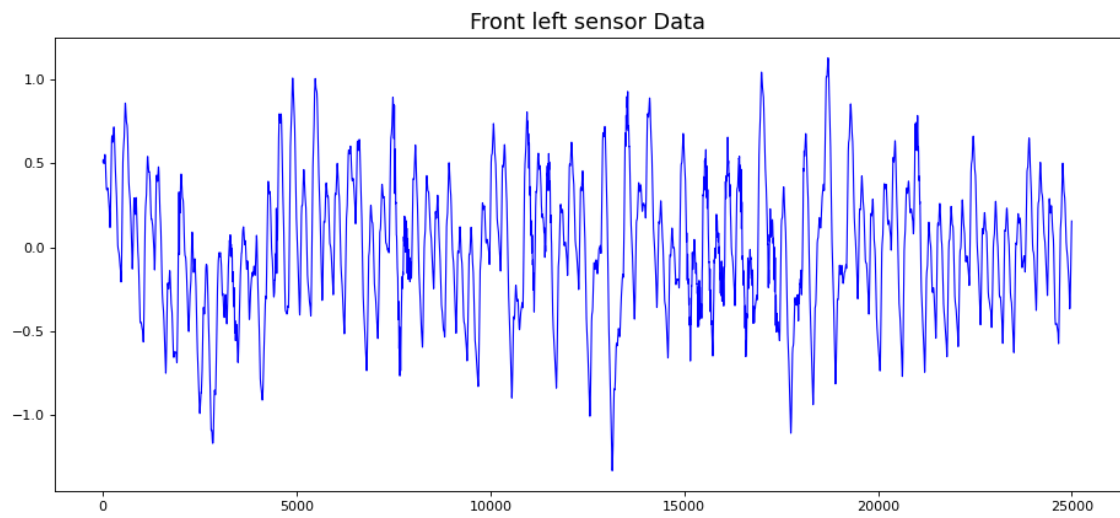


Figure 8 - Sensor positioned in the front left

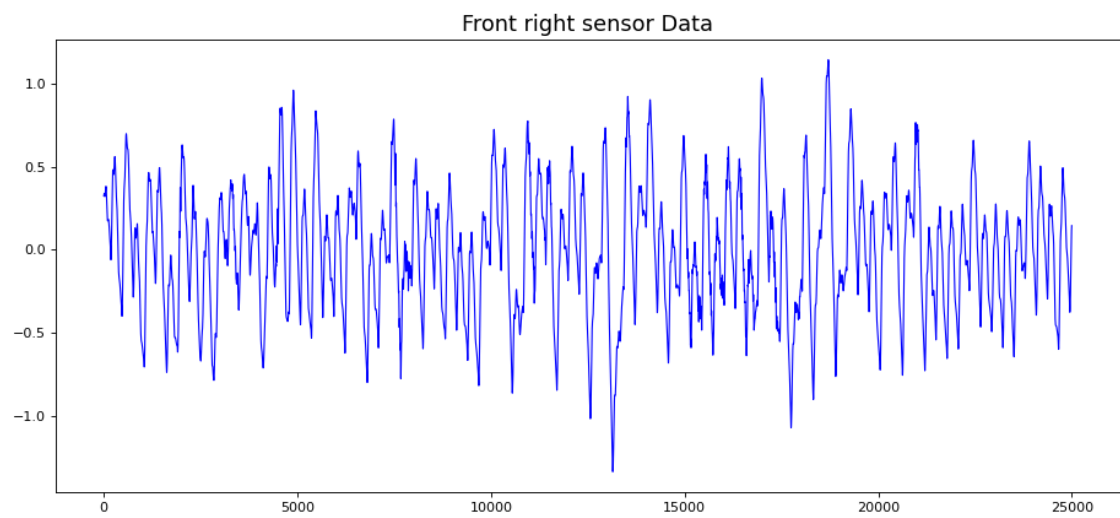


Figure 9 - Sensor positioned in the front right

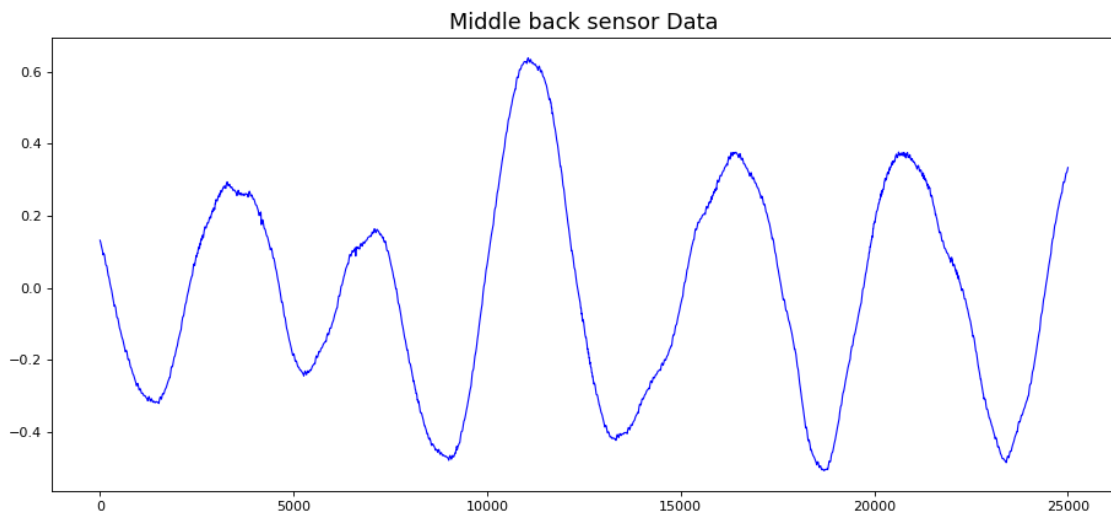


Figure 10 - Sensor positioned in the middle back

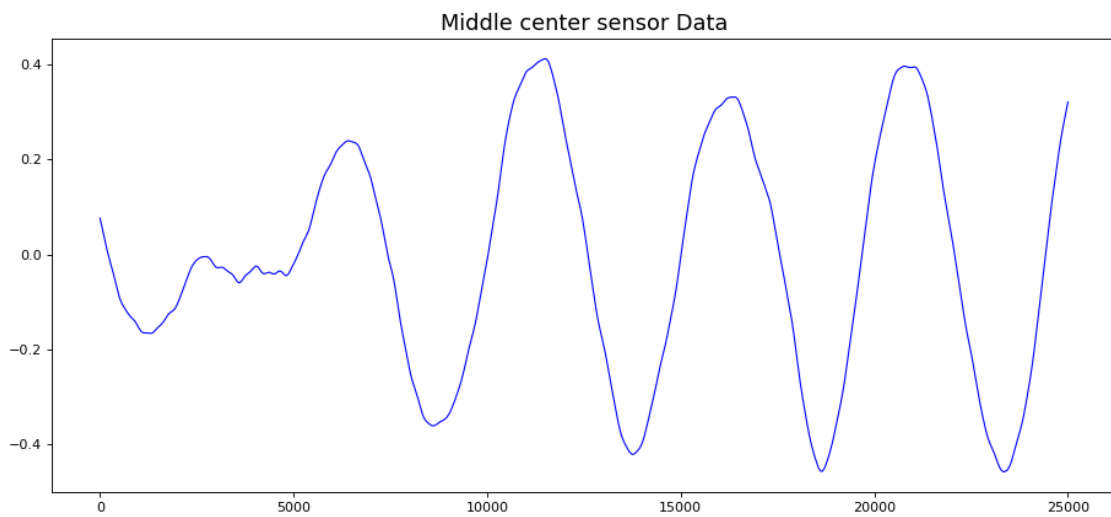


Figure 11 - Sensor positioned in the middle center

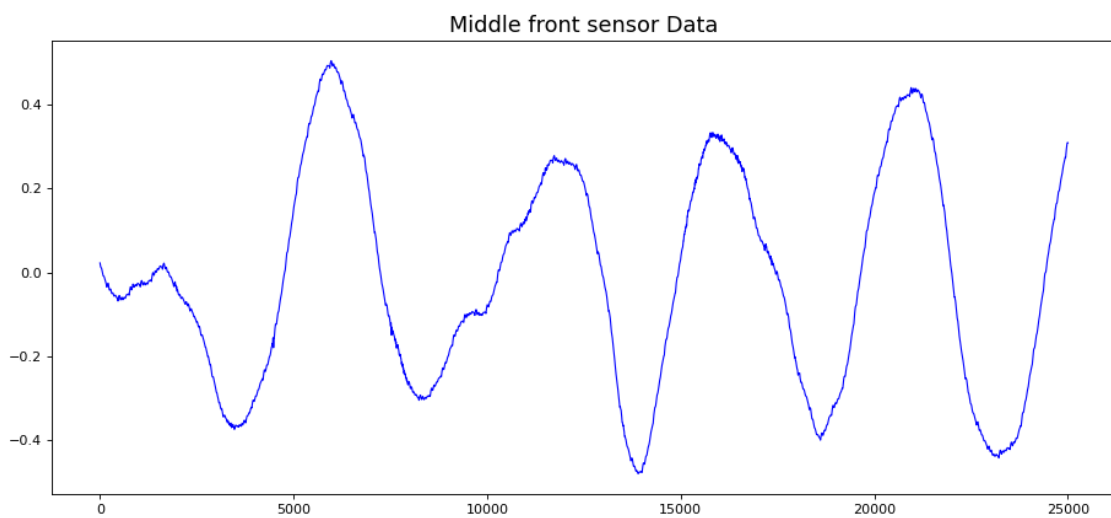


Figure 12 - Sensor positioned in the middle front

At the present date, only two real anomalies are provided with detail and represented in defect scenarios. The defects are applied to the vertical irregularity of the left rail. The following figures illustrate the different anomalies:

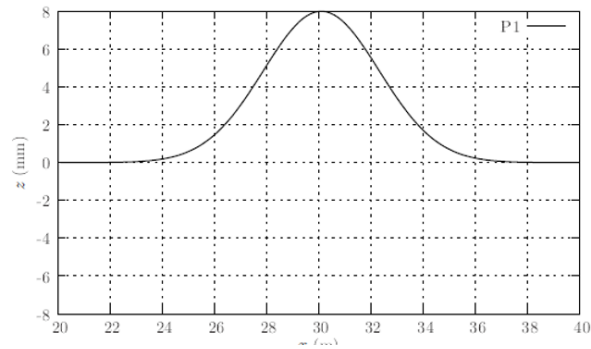


Figure 13 - Anomaly P1

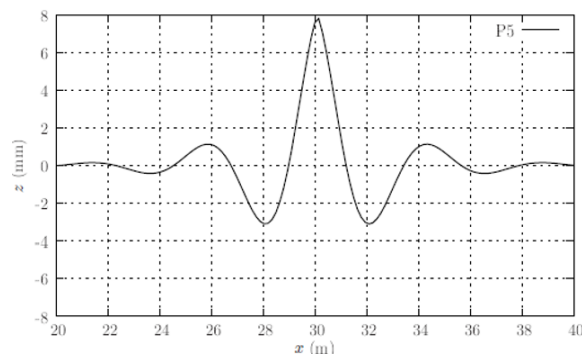


Figure 14 - Anomaly P5

To give a clearer perspective of the representation of an anomaly, it is possible to see how the anomaly P1 with the greatest amplitude present in the dataset diverges from the normal data in the following images. The blue line represents normal data and the red line the anomalous data. Figure 15 shows the difference in the front left sensor, while Figure 16 the back left sensor. It is possible to easily identify the anomaly. The difference in distance from where the anomaly is detected in each picture is related to the distance between the two sensors.

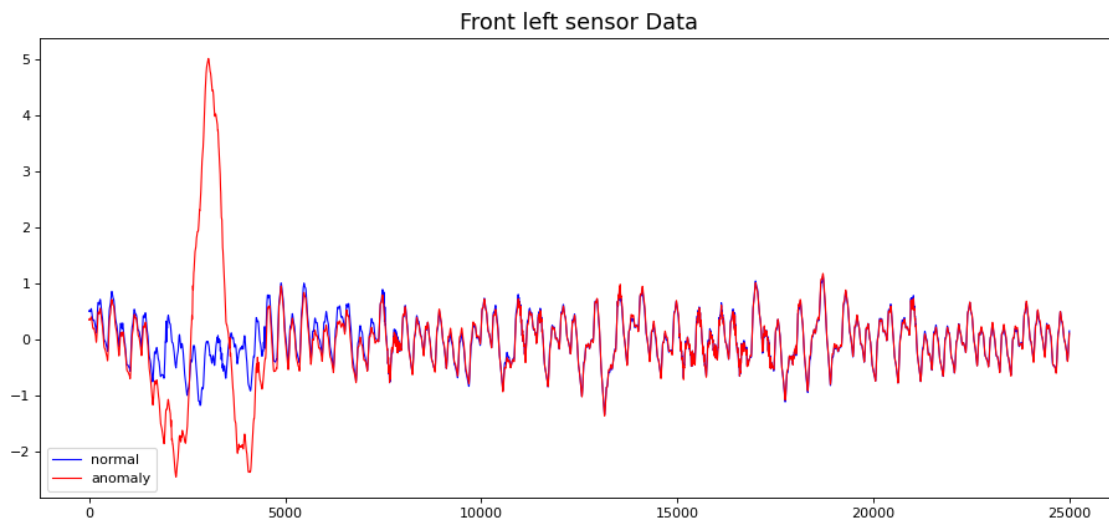


Figure 15 - Normal vs anomalous data in the front left sensor

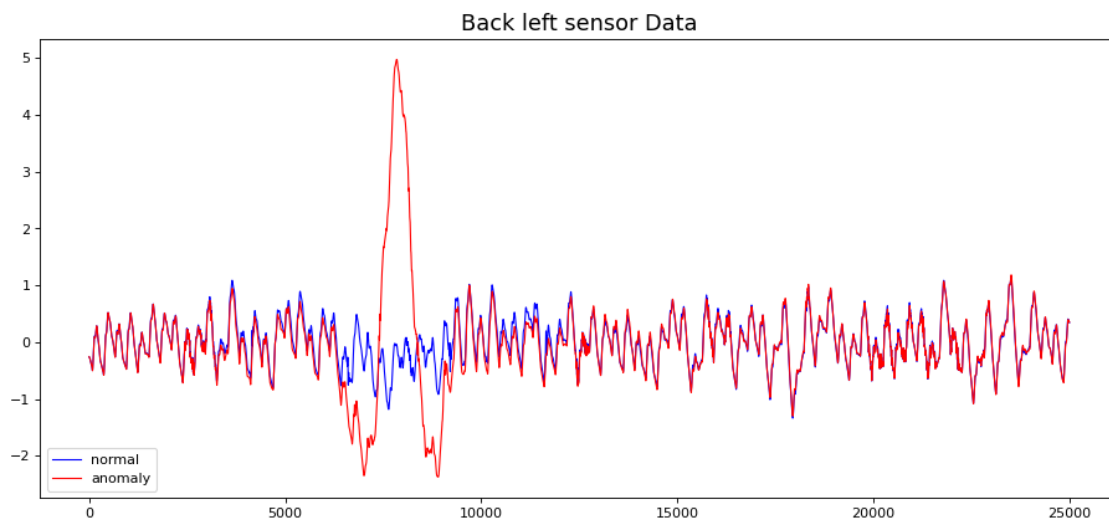


Figure 16 – Normal vs anomalous data in the back left sensor

Different defect amplitudes are present in the dataset: 6 millimeters, 8 millimeters, 10 millimeters, and 17 millimeters. Since the amplitudes can be either positive or negative, 8 profiles were taken into account. Each amplitude corresponds to an anomaly level:

- Alert – 8 millimeters
- Intervention – 10 millimeters
- Immediate action - 18 millimeters

Furthermore, both anomalies simulated occur always in the same place, that is, at the same distance from the beginning of the departure point.

### **3.3.2 Privacy and Ethical Considerations**

This project does not present any ethical considerations, since all the data is generated from mathematical equations. On the other hand, the dataset must follow a privacy agreement, and must not be disclosed.



## 4 Experimentation and Validation

This chapter presents the evaluation metrics used to determine the performance of the models tested and the pre-processing and processing experimentations to determine the most suited models to the problem at hands.

### 4.1 Data used for training and test

The data provided for this project, as already described in the previous chapter, allowed the author to analyze different scenarios and approaches for delivering a model capable of detecting anomalies. During this phase, a few things were taken into consideration to decide how to train the models.

The first thing to consider was the difference in size for each dataset that corresponded to different train speeds. If a train goes at 50 kilometers per hour and another goes at 70 kilometers per hour, the first will have more records, since the traveled distance is the same, but it will take more time to travel that distance. The variation in sizes proved to be a challenge since the idea was to train the models with the whole sequence or sub-sequences, and this way it would be more difficult to guarantee that the input data had the correct length. Therefore, it was decided that training a model for each train speed would be the most viable solution.

Furthermore, since the provided data only has anomaly scenarios for a train going at 75 kilometers per hour, the decision was to use a baseline scenario with that train speed for training. Moreover, the anomaly scenarios provided only address one irregularity profile, consequently, it shall be used for training the normal scenario that has that irregularity. Also, each scenario had 25001 rows, and if more than one scenario was used, there were hardware limitations upon training the models.



## 4.2 Pre-processing data

In this step of the project, not many things needed to be done regarding data pre-processing. Since the data was manually generated, it did not present any null values or outliers that required intervention.

Furthermore, the sensor data is only numerical values, so there were no categorical features to consider. The only step taken was to normalize all the sensor data so that the values of each sensor were between 0 and 1. This way, there are no scale differences, numerical instability is avoided, and the data is more easily interpreted. The scaler used to achieve this was MinMax. This scaler subtracts the minimum value from each data point and divides it by the range, which is the difference between the maximum and minimum values.

Finally, the column time was dropped, and only the sensor data were used as features, thus having 7 features.

## 4.3 Anomaly Detection Tested Models

For this project, it was decided to test the machine learning model OC-SVM, and two deep learning approaches by using autoencoders. The reason for this is that these technologies are the most widely used, as seen in the State-of-the-Art chapter, and more promising. For each model, it was used the same set of data for training (one baseline scenario with 25001 rows) and the same data for testing (more baseline scenarios and anomaly scenarios). For the deep learning models, the KerasTuner was used to find the best hyperparameters. Each training step consisted of 50 epochs, and the batch size used was 64.

### 4.3.1 One-Class SVM

The one-class SVM algorithm used was provided by Scikit-Learn. The code for defining the model is provided in the following code snippet:

```
def oneclass_svm(data):  
    model = OneClassSVM(nu=0.005, kernel='rbf', gamma='scale')  
    model.fit(data)  
    return model  
  
model = oneclass_svm(X_train)
```

Code Snippet 1 – One-Class SVM model

The code previously shown has a few considerations. From the research made, it is not possible to use a tuner to obtain the best hyperparameters for an OC-SVM using only normal data,

therefore, the tuning process was made by hand for this model. The set of hyperparameters chosen to tune were the following:

- **nu** – It is a value between 0 and 1 and corresponds to an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors, which means that it represents the approximate fraction of outliers expected in the data, controlling the trade-off between the training error and the ability to capture outliers.
- **Kernel** – Specifies the kernel function to be used. In simple terms, the kernel calculates the similarity measure between two data points, by mapping the original input data into a higher dimensional space, where patterns or classes can be more distinguishable [46]. A few kernel functions are available for OC-SVM (linear, poly, rbf, sigmoid, precomputed).
- **Gamma** – It is the kernel coefficient for “rbf”, “poly”, and “sigmoid”. It controls the influence of individual training samples, meaning that higher values lead to a more complex decision boundary, and can result in overfitting, while lower values make the decision boundary smoother [47]. It can take the values “scale”, “auto”, or a float. If it is “scale”, then the value is 1 divided by the number of features multiplied by the variance of the input data. If “auto” is chosen, then the coefficient is 1 divided by the number of features. Finally, if it is a float, then it must be a positive value.

The following table resumes the best hyperparameters found for this model:

Table 4 – Best hyperparameters for OC-SVM

Hyperparameter name	Value
“nu”	0.005
“Kernel”	rbf
“Gamma”	scale

With the model trained, to make new predictions the only step required is to call the “predict” function.

#### 4.3.2 Convolutional Autoencoder

One of the tested models for this project was a convolutional reconstruction autoencoder model. Convolutional layers are most commonly used for image processing and computer vision given their efficiency for those use cases, but they can also be used in other cases, like anomaly detection in time series. Unlike traditional machine learning methods that require diverse techniques of preprocessing, the advantage of Convolutional Neural Networks (CNN) is that

they can automatically learn relevant features from the data. To address data that comes from signals, the most adequate approach is to use 1 dimension convolutional layers [48].

To have a better understanding, a convolutional layer computes the input data and outputs feature maps that are introduced to the next layer. Since the second and subsequent layers operate with the output of the previous layers, the more the number of layers, the more abstract features will the network be able to learn. Each convolutional layer is composed of kernels. These kernels convolve the input data, which generates a feature map as output. Note that the kernels and input are represented as multidimensional arrays. The following picture illustrates the data in a 1-dimensional convolution [49].

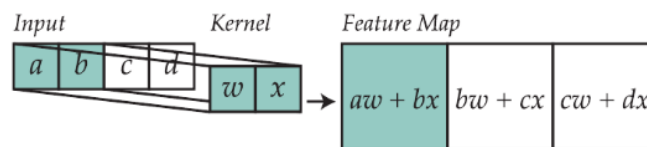


Figure 17 – 1D convolution example

This model required adapting the data beforehand and reshaping it. The input shape of the model should be (batch size, sequence length, and number of features), and the output should be of the same shape.

Two versions of this model will be tested. One will have a sequence length (time steps) equal to 1, meaning that the autoencoder processes a single snapshot of data with multiple features at a time, without considering any temporal dependencies or sequential patterns. The second version will have sequences of 20, this way having the capability to capture temporal dependencies and patterns.

The code for creating the training sequences is the following (in this case for 20 timesteps) (24982, 20, 7):

```
TIME_STEPS = 20
def create_sequences(values, time_steps=TIME_STEPS):
    output = []
    for i in range(len(values) - time_steps + 1):
        output.append(values[i : (i + time_steps)])
    return np.stack(output)

x_train = create_sequences(df_training_value)
```

Code Snippet 2 – Training sequences for convolutional reconstruction autoencoder model

Note that the value for timesteps is 20 because it was considered an adequate interval to feed subsequences to the model. Furthermore, if this value was increased more, the training time would take a lot more time and possibly end with memory overflow. The shape of the data for 1 timestep and 20 timesteps is (25001, 1, 7) and (24982, 20, 7), respectively. The 24982 value

is obtained because it is not possible to create another sequence of 20 with the remaining data points.

Having the correct input shape in hands, it is possible to construct and train the model. The following code snippet shows the model implementation, which once again is based on KerasTuner in order to find the best hyperparameters.

```
def build_model(hp):
    model = keras.Sequential(
        [
            layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
            layers.Conv1D(
                filters=hp.Int('filters', 32, 128, 16),
                kernel_size=hp.Choice('kernel_size', [3, 5, 7]), padding="same", strides=2,
                activation="relu"
            ),
            layers.Dropout(rate=hp.Float('dropout_rate', 0.0, 0.5, 0.1)),
            layers.Conv1D(
                filters=hp.Int('filters', 32, 128, 16),
                kernel_size=hp.Choice('kernel_size', [3, 5, 7]), padding="same", strides=2,
                activation="relu"
            ),
            layers.Conv1DTranspose(
                filters=hp.Int('filters', 32, 128, 16),
                kernel_size=hp.Choice('kernel_size', [3, 5, 7]), padding="same", strides=2,
                activation="relu"
            ),
            layers.Dropout(rate=hp.Float('dropout_rate', 0.0, 0.5, 0.1)),
            layers.Conv1DTranspose(
                filters=hp.Int('filters', 32, 128, 16),
                kernel_size=hp.Choice('kernel_size', [3, 5, 7]), padding="same", strides=2,
                activation="relu"
            ),
            layers.Conv1DTranspose(filters=7, kernel_size=7, padding="same"),
        ]
    )
    learning_rate = hp.Choice('learning_rate', [0.1, 0.01, 0.001])
    model.compile(optimizer=keras.optimizers.Adam(learning_rate=learning_rate),
                  loss="mse")

    return model
```

Code Snippet 3 – Convolutional reconstruction autoencoder model

In Code Snippet 3 the function to obtain the model is defined. The first step is to use Keras “sequential”, which allows for defining a sequence of layers for the model. The convolutional layers will act as the encoders, while the transposed convolutional layers will act as decoders, reconstructing the data. The model layers are the following:

1. "layers.Input" – Entry point for the model
2. "layers.Conv1D" – First convolutional layer
3. "layers.Dropout" – Regularization layer that applies a dropout preventing overfitting
4. "layers.Conv1D" – Second convolutional layer
5. "layers.Conv1DTranspose" – First transposed convolutional layer
6. "layers.Dropout" – Another dropout layer
7. "layers.Conv1DTranspose" – Second transposed convolutional layer
8. "layers.Conv1DTranspose" – Final transposed convolutional layer without an activation function

For this model, the following hyperparameters are considered:

- `learning_rate` – Controls the step size at which the model's parameters are updated during the training process using optimization algorithms. Commonly, this value ranges between 0.1 and 0.0001, but it might vary depending on the optimizer used.
- `filters` – Represents the number of output filters in the convolution. The value must be an integer.
- `kernel_size` – Specifies the length of the 1-dimensional convolution window. It can be an integer or a tuple/list of a single integer.
- `dropout_rate` – Specifies the proportion of random neurons that will be deactivated or set to 0 in each training step, thus preventing overfitting. This value can vary between 0 and 1, but most commonly is equal to or less than 0.1.

Furthermore, the optimization algorithm chosen for updating the parameters of the neural network was Adam, and the activation function, commonly used for CNNs, used was ReLU (Rectified Linear Unit).

For model training and hyperparameter tuning, the process is the following:

```
tuner = keras_tuner.RandomSearch(
    hypermodel=build_model,
    objective="loss",
    max_trials=10,
)
# Run the hyperparameter search
tuner.search(x_train, x_train, epochs=50, batch_size=64)

# Get the best hyperparameters
best_hp = tuner.get_best_hyperparameters()[0].values
```

Code Snippet 4 – Convolutional AutoEncoder training

In – Convolutional AutoEncoder training Code Snippet 4, the tuner is constructed. Here, a random search is used. RandomSearch Keras Tuner is a hyperparameter tuning technique that uses random sampling to search for optimal hyperparameter values for the model. It selects combinations of hyperparameters randomly from ranges that are predefined, trains and

evaluates the model with each combination, and is capable of identifying the best hyperparameter. This approach is computationally less expensive than a grid search, helping to find the hyperparameter values in a shorter amount of time. In the code, the “hypermodel” field is the model to train, the “objective” is the metric used to evaluate the performance of each model, and the “max\_trials” is the total number of trials to test different model configurations. Again, since the aim is to train the model with normal data, the “search” function only receives the input features.

The following tables resume the best hyperparameters found for this model:

Table 5 – Best hyperparameters for convolutional AE with 1 timestep

Hyperparameter name	Value
“filters”	64 (same in every layer)
“kernel_size”	7 (same in every layer)
“dropout_rate”	0.0
“learning_rate”	0.001

Table 6 – Best hyperparameters for convolutional AE with 20 timesteps

Hyperparameter name	Value
“filters”	48 (same in every layer)
“kernel_size”	5 (same in every layer)
“dropout_rate”	0.0 confirm if not 0.2
“learning_rate”	0.01

To make new predictions with an autoencoder, it is necessary to determine a threshold first. To achieve this, it is necessary to calculate the reconstruction error (e.g. Mean Squared Error) between the input data (training data) and the output reconstructed by the autoencoder. Then, the maximum value of the loss will be taken as the threshold, since it is the worst the model has performed trying to reconstruct a sample. If the reconstruction loss for a sample is greater than the threshold, then the model is seeing a pattern that it is not familiar with. Consequently, this sample will be labeled as an anomaly.

Code Snippet 5 exposes this process, by first predicting the training values, followed by a calculation of the Mean Absolute Error (MAE) and calculating the threshold. The thresholds

obtained for the AE with 1 timestep and for the AE with 20 timesteps, were 0.60823 and 0.29655, respectively.

```
X_pred = model.predict(X_train)
scored = pd.DataFrame(index=df_train.index[0:25001])
scored['Loss_mae'] = np.mean(np.abs(X_pred - X_train), axis=(1, 2))
threshold = np.max(scored['Loss_mae'])
```

Code Snippet 5 – Calculating the threshold

Now, for making new predictions and determining if there are anomalies in the data, the process is to compare the reconstruction error of the test samples with the threshold. Code Snippet 6 exposes how to implement this code, and Figure 18 is an example of an anomaly profile reconstructed by the trained autoencoder.

```
df_test = pd.read_csv('dataset_10_P5.csv')
df_test = df_test.drop(columns=['Time'])

x_test = create_sequences(df_test.values)

# Get test MAE loss.
X_pred = best_model.predict(x_test)

scored = pd.DataFrame(index=df_test.index[0:25001])

scored['Loss_mae'] = np.mean(np.abs(X_pred - x_test), axis=(1, 2))

scored['Threshold'] = threshold

scored['Anomaly'] = scored['Loss_mae'] > scored['Threshold']
```

Code Snippet 6 – AE predictions on the test set

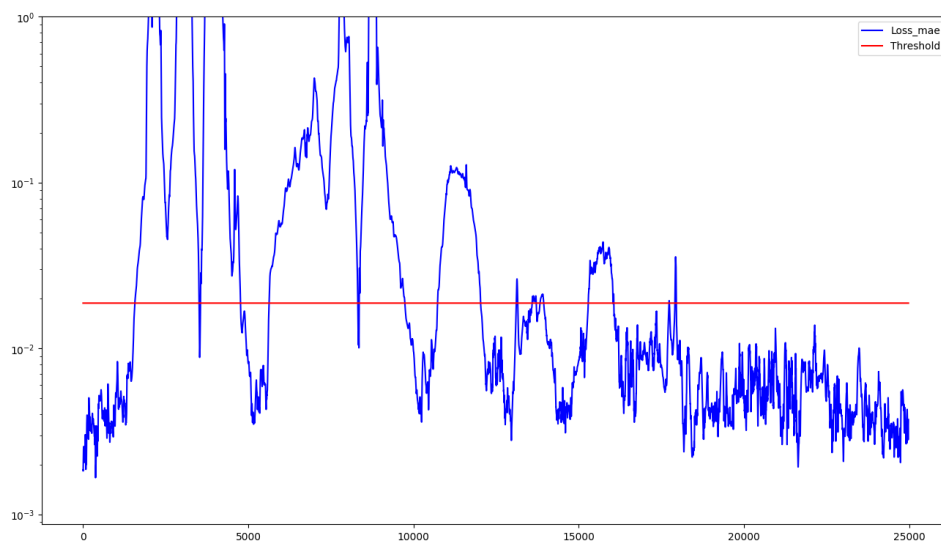


Figure 18 – MAE vs threshold for an anomaly scenario

### 4.3.3 LSTM Autoencoder

LSTM is a specific type of recurrent neural network that aims to model temporal sequences and their long-term dependencies more accurately than common recurrent neural networks, which can be benefic to time-series problems. The traditional LSTM architecture has memory blocks with self-connections and gates to control the information flow. The original architecture had input and output gates; later, a forget gate was added to process continuous input streams. Modern LSTM also includes peephole connections for precise output timing [50].

The first step done to train an LSTM autoencoder was to reshape the data to the correct input format (samples, timesteps, and features). Similar to the previous model.

After, it was time to define a function to build the model. The following code snippet represents that function:

```
def build_model(hp):
    X=X_train
    inputs = Input(shape=(X.shape[1], X.shape[2]))
    L1 = LSTM(hp.Int('layer_1_neurons',min_value=4,max_value=256,step=32),
activation=hp.Choice('dense_activation',values=['relu', 'sigmoid']),
return_sequences=True,
kernel_regularizer=regularizers.l2(0.00))(inputs)
    L2 = LSTM(hp.Int('layer_2_neurons',min_value=4,max_value=256,step=32),
activation=hp.Choice('dense_activation',values=['relu', 'sigmoid']),
return_sequences=False)(L1)
    L3 = RepeatVector(X.shape[1])(L2)
    L4 = LSTM(hp.Int('layer_3_neurons',min_value=4,max_value=256,step=32),
activation=hp.Choice('dense_activation',values=['relu', 'sigmoid']),
return_sequences=True)(L3)
    L5 = LSTM(hp.Int('layer_4_neurons',min_value=4,max_value=256,step=32),
activation=hp.Choice('dense_activation',values=['relu', 'sigmoid']),
return_sequences=True)(L4)
    output = TimeDistributed(Dense(X.shape[2]))(L5)
    model = Model(inputs=inputs, outputs=output)
    learning_rate = hp.Choice('learning_rate', [0.1, 0.01, 0.001])
    model.compile(optimizer=keras.optimizers.Adam(learning_rate=learning_rate),
loss='mae')
    return model
```

Code Snippet 7 – LSTM Autoencoder model

The model has five layers:

1. L1 – LSTM layer, corresponds to the encoder and returns a compressed sequence of the input
2. L2 – LSTM layer which acts as the latent space, where the input sequence is reduced to a fixed-length representation



3. L3 – Repeat vector layer, that repeats the output of the latent space to have the same number of time steps as the input sequence, preparing this way the data for the decoder
4. L4 – LSTM layer that takes the output of the previous layer and generates a sequence of hidden states, preparing it for the final decoding step
5. L5 – The last LSTM layer that processes the sequences of the hidden states and tries to reconstruct the input sequence at its output
6. Final layer – Time distributed dense layer, serves as the output layer that applies a dense layer to each time step of the reconstructed sequences

For this model, the following hyperparameters are considered:

- layer\_1\_neurons – Number of neurons for L1. It is an integer that ranges from 4 to 256, having a step size of 32. It controls the capacity and complexity of the first LSTM layer.
- layer\_2\_neurons – Same as the previous hyperparameter, but for L2. It controls the capacity and complexity of the latent space layer.
- layer\_3\_neurons – Same as the previous hyperparameter, but for L4. It controls the capacity and complexity of the first decoder layer.
- layer\_4\_neurons – Same as the previous hyperparameter, but for L5. It controls the capacity and complexity of the second decoder layer.
- dense\_activation – Activation function used in the LSTM layers, which determines the non-linearity in the LSTM cells. Categorical hyperparameter, with the options “relu” or “sigmoid”.
- dropout\_rate – As already explained in the previous model, specifies the proportion of random neurons that will be deactivated or set to 0 in each training step, thus preventing overfitting. This value can vary between 0 and 1, but most commonly is equal to or less than 0.1.

Furthermore, the optimization algorithm chosen for updating the parameters of the neural network was Adam.

The step of training and finding the best hyperparameters is similar to the previous autoencoder, demonstrated in Code Snippet 4.

The following tables resume the best hyperparameters found for the model:

Table 7 – Best hyperparameters for LSTM AE with 1 timestep

Hyperparameter name	Value
"layer_1_neurons"	132
"layer_2_neurons"	228
"layer_3_neurons"	132
"layer_4_neurons"	100
"dense_activation"	relu
"learning_rate"	0.01

Table 8 - Best hyperparameters for LSTM AE with 20 timesteps

Hyperparameter name	Value
"layer_1_neurons"	68
"layer_2_neurons"	164
"layer_3_neurons"	228
"layer_4_neurons"	164
"dense_activation"	relu
"learning_rate"	0.001

Once again, to make new predictions with this autoencoder, it is required to set the threshold. This process is the same as the previous model.

## 4.4 Validation Metrics

This section describes the validation metrics used in this work.

### 4.4.1 Anomaly Detection Algorithms Metrics

Although the used algorithms address an OCC problem, the testing phase involves using anomalous and non-anomalous data. Therefore, the evaluation metrics used can be the ones applied to supervised classification problems.

#### 4.4.1.1 Confusion Matrix

Generally, the evaluation metrics for supervised classification problems are computed based on the Confusion Matrix. In this matrix, the input class labels (real values) are represented in the rows, and the output class labels (predicted values) are represented in the columns [51].

In binary classification problems, a confusion matrix typically has four entries [52]:

- True Positives (TP) – Instances correctly classified as positive
- False Positives (FP) – Instances incorrectly classified as positive
- True Negatives (TN) – Instances correctly classified as negative
- False Negatives (FN) – Instances incorrectly classified as negative

The following picture illustrates the confusion matrix:

True Class	Predicted Class	
	True	False
True	True Positive (TP)	False Negative (FN)
False	False Positive (FP)	True Negative (TN)

Figure 19 – Confusion Matrix representation [53]

#### 4.4.1.2 Accuracy

Accuracy is a metric that provides the proportion of the total number of correct predictions. Its value varies between 0 and 1, 0 meaning that no prediction was correct and 1 meaning that all predictions were correct [52]. It can be calculated with the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

#### 4.4.1.3 Precision and Recall

Precision, or positive predictive value, measures the proportion of positive instances that were correctly classified as positive among all the instances classified as positive [52]. It can be calculated with the following formula:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

On the other hand, Recall, also known as Sensitivity or True Positive Rate, measures the proportion of positive instances that were correctly classified as positive [52]. It can be calculated with the following formula:

$$Precision = \frac{TP}{TP + FN} \quad (3)$$

#### 4.4.1.4 F1 Score

F1-score is the harmonic mean of precision and recall and provides importance to both factors [52]. It can be calculated with the following formula:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

### 4.4.2 Models performance

Having an understanding of the evaluation metrics to use, the next step is to evaluate the performance of the models trained.

To evaluate the model performance, different scenarios will be tested:

- 3 scenarios with anomaly P1, each with a different anomaly severity (low, medium, high)
- 3 scenarios with anomaly P5, each with a different anomaly severity (low, medium, high)
- 2 baseline scenarios with different irregularities

Having the metrics of each scenario, the last step will be to calculate an average and obtain a final score.

#### 4.4.2.1 OC-SVM

Table 9 summarizes the performance of the OC-SVM model. For the baseline scenarios (with no anomalies), the model has an overall good performance. For the detection of anomalies, it is possible to see that it has high precision, meaning that the true positives that it detect are mostly correct. On the other hand, the recall is a little lower, which means the model is prone to find less true positives. Looking at the accuracy, the model presents an overall correctness of its predictions.

Table 9 – OC-SVM metrics per scenario

Scenarios	Precision	Recall	Accuracy	F1-score
1 <sup>st</sup> baseline scenario	1	0.97	0.97	0.98
2 <sup>nd</sup> baseline scenario	1	0.97	0.97	0.98
Anomaly P1 low severity	0.85	0.88	0.79	0.86
Anomaly P1 medium severity	0.98	0.52	0.63	0.68
Anomaly P1 high severity	0.96	0.74	0.78	0.84
Anomaly P5 low severity	0.99	0.45	0.58	0.62
Anomaly P5 medium severity	0.99	0.77	0.82	0.87
Anomaly P5 high severity	0.99	0.87	0.89	0.92
<b>Average</b>	0.97	0.77	0.80	0.84

#### 4.4.2.2 CNN-AE

##### **CNN-AE with 1 timestep**

The performance of the CNN-AE model with 1 timestep is summarized in Table 10. For the baseline scenarios, the model has an excellent performance. For the detection of anomalies, the precision is the same as the previous model. On the other hand, the ability to detect anomalies is lower than in the OC-SVM model, but the accuracy is greater.

Table 10 – CNN-AE model with 1 timestep, metrics per scenario

Scenarios	Precision	Recall	Accuracy	F1-score
1 <sup>st</sup> baseline scenario	1	1	1	1
2 <sup>nd</sup> baseline scenario	1	0.99	0.99	0.99
Anomaly P1 low severity	0.93	0.05	0.77	0.10
Anomaly P1 medium severity	0.92	0.61	0.89	0.74
Anomaly P1 high severity	0.95	0.52	0.88	0.68
Anomaly P5 low severity	0.99	0.31	0.83	0.47
Anomaly P5 medium severity	0.99	0.90	0.90	0.95
Anomaly P5 high severity	0.99	0.93	0.93	0.96
<b>Average</b>	0.97	0.66	0.90	0.74

### CNN-AE with 20 timesteps

The performance of the CNN-AE model with 20 timesteps is summarized in Table 11. For the baseline scenarios, the model has an excellent performance, as same as the model with 1 timestep. For the detection of anomalies, the precision is lower than the CNN-AE with 1 timestep. Oppositely, the recall is greater. The accuracy is lower, and the F1-score is identical.

Table 11 - CNN-AE model with 20 timesteps, metrics per scenario

Scenarios	Precision	Recall	Accuracy	F1-score
1 <sup>st</sup> baseline scenario	1	0.99	0.99	0.99
2 <sup>nd</sup> baseline scenario	1	0.99	0.99	0.99
Anomaly P1 low severity	0.93	0.60	0.89	0.74
Anomaly P1 medium severity	0.52	0.79	0.77	0.62
Anomaly P1 high severity	0.60	0.78	0.82	0.68
Anomaly P5 low severity	0.42	0.86	0.68	0.56
Anomaly P5 medium severity	0.50	0.83	0.76	0.62
Anomaly P5 high severity	0.45	0.93	0.71	0.60
<b>Average</b>	0.67	0.85	0.83	0.73

#### 4.4.2.3 LSTM-AE

##### **LSTM-AE with 1 timestep**

The performance of the LSTM-AE with 1 timestep model is summarized in Table 12. For the baseline scenarios, the model has an excellent performance. It has good precision, but the recall is poor in comparison with the models already analyzed, also reflected in the F1-score. Finally, the accuracy is decent.

Table 12 - LSTM-AE model with 1 timestep, metrics per scenario

Scenarios	Precision	Recall	Accuracy	F1-score
1 <sup>st</sup> baseline scenario	1	1	1	1
2 <sup>nd</sup> baseline scenario	1	1	1	1
Anomaly P1 low severity	1	0.02	0.76	0.04
Anomaly P1 medium severity	0.65	0.47	0.81	0.55
Anomaly P1 high severity	1	0.06	0.77	0.11
Anomaly P5 low severity	0.75	0.57	0.85	0.66
Anomaly P5 medium severity	0.99	0.20	0.80	0.33
Anomaly P5 high severity	0.99	0.03	0.76	0.07
<b>Average</b>	0.92	0.42	0.84	0.47

### LSTM-AE with 20 timesteps

The performance of the LSTM-AE model with 20 timesteps is summarized in Table 13. For the baseline scenarios, the model has an excellent performance. It has a lower precision in comparison with the previous models, but the recall outstands among them. The accuracy and F1-score have reasonable values.



Table 13 - LSTM-AE with 20 timesteps, metrics per scenario

Scenarios	Precision	Recall	Accuracy	F1-score
1 <sup>st</sup> baseline scenario	1	0.99	0.99	0.99
2 <sup>nd</sup> baseline scenario	1	0.99	0.99	0.99
Anomaly P1 low severity	0.63	0.75	0.84	0.89
Anomaly P1 medium severity	0.43	0.99	0.68	0.60
Anomaly P1 high severity	0.60	0.97	0.84	0.74
Anomaly P5 low severity	0.37	0.99	0.59	0.54
Anomaly P5 medium severity	0.55	0.99	0.80	0.71
Anomaly P5 high severity	0.59	0.99	0.83	0.74
<b>Average</b>	0.65	0.96	0.82	0.78

#### 4.4.2.4 Final remarks

To conclude the overview of the performance of the models tested, the following table displays the average of each metric for the models tested:

Table 14 – Overview of the models performance

Model	Precision	Recall	Accuracy	F1-score
OC-SVM	<b>0.97</b>	0.77	0.80	<b>0.84</b>
CNN-AE 1 timestep	<b>0.97</b>	0.66	<b>0.90</b>	0.74
CNN-AE 20 timesteps	0.67	0.85	0.83	0.73
LSTM-AE 1 timestep	0.92	0.42	0.84	0.47
LSTM-AE 20 timesteps	0.65	<b>0.96</b>	0.82	0.78

In summary, the OC-SVM and CNN-AE with 1 timestep are the models with better precision, therefore they present more correctness in detecting anomalies in anomalous scenarios, and normality in scenarios where anomalies are not present. Furthermore, CNN-AE with 1 timestep has the overall best accuracy, which means it makes more correct predictions on the majority of data, and the OC-SVM is the model that has the best F1-score. In contrast, the LSTM-AE with 20 timesteps model is the one that outstands in the recall measurement, meaning that, among

the models displayed, it is the better at capturing anomalies in anomalous scenarios, and normality in normal scenarios, although for the last case, all the models performed identically.

The model chosen to develop a proof of concept to detect and classify anomalies in a railway track was the LSTM-OC with 20 timesteps. The reason is that it has adequate precision and accuracy, although other models outstand it. On the other hand, since it is an LSTM and has timesteps, it might be capturing anomalous patterns that were not labeled by the author. Furthermore, a high recall is important in a PdM, since having more false negatives can lead to not detecting anomalies with catastrophic consequences (e.g. derailment).

## 4.5 Anomaly classification Approach

Since the model chosen to detect anomalies was an autoencoder, the approach to classify the severity of the anomalies was to use the thresholds provided by that model as anomaly scores. With these anomaly scores, it is possible to set anomaly levels.

In greater detail, the process to define these anomaly levels was the following:

1. Calculation of the threshold for each of the 10 anomaly scenarios provided in the dataset.
2. Ordering of the thresholds (anomaly scores) by anomaly level (from lowest to greatest).
3. Use of percentiles to obtain 3 anomaly severity levels:
  - a. Scenarios where the anomaly score is below the 25<sup>th</sup> percentile are considered of low severity.
  - b. Scenarios where the anomaly score is between the 25<sup>th</sup> and 75<sup>th</sup> percentile are considered of medium severity.
  - c. Scenarios where the anomaly score is higher than the 75<sup>th</sup> percentile are considered high severity.



# 5 Prototype implementation

This section describes the implementation of the prototype to detect and classify anomalies, as described in Chapter 3.

## 5.1 Service Bus

Azure service bus is a cloud-based message service, that allows changing information between applications and services. It provides messaging capabilities such as queuing and publish/subscribe patterns, and in this case, a publish/subscribe pattern will be employed.

Figure 20 displays an example of sensor data in the service bus.

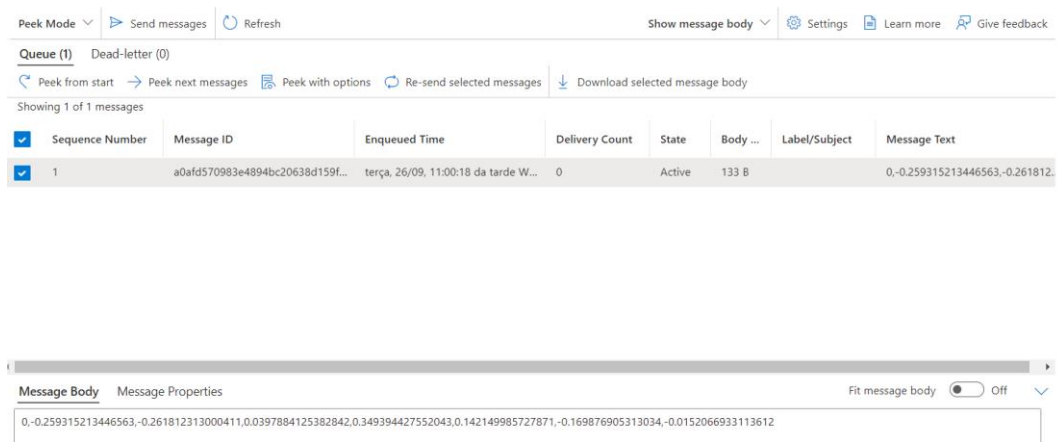


Figure 20 - Service bus message example

## 5.2 Azure Function

The implemented Azure function is connected to the Service bus in Azure. The Azure function will subscribe to that service bus and read a batch of data. In this case, since the model processes the data 20 timesteps at a time, it will subscribe to batches of 20. Then, it will post that data to the anomaly detection and classification modules.

## 5.3 Anomaly Detection and classification Modules

This module uses the LSTM autoencoder model with 20 timesteps, trained in the previous chapter. A POST endpoint is provided to test the flow of detecting and classifying anomalies in the railway track. The following code snippet displays the implementation:

```
import requests
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.get_json(force=True)
        if not isinstance(data, list):
            return jsonify({"error": "Invalid data format. Expected a list of rows."}), 400

        X_test = create_sequences(data)
        X_pred = model.predict(X_test)
        scored = pd.DataFrame(index=data.index)
        scored['Loss_mae'] = np.mean(np.abs(X_pred - data), axis=(1,2))

        if(scored['Loss_mae'].max() >= high_severity_threshold):
            return jsonify({"message": "High severity anomalies in the track!"}), 200
        elif(scored['Loss_mae'].max() >= medium_severity_threshold):
            return jsonify({"message": "Medium severity anomalies in the track!"}), 200
        elif(scored['Loss_mae'].max() >= low_severity_threshold):
            return jsonify({"message": "Low severity anomalies in the track!"}), 200

        return jsonify({"message": "No anomalies detected."}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    model_api_url = 'https://anomalydetection.com/predict'
```

```
app.run(host='0.0.0.0', port=5000)
```

Code Snippet 8 – Endpoint to detect and classify anomalies

In this example, the endpoint will receive the dispatched data from the Azure function as a batch of 20 elements. This batch will be converted to the correct input format, and the anomaly detection model will predict if any anomalies exist. Then, instead of comparing to a single threshold the mean absolute error, the maximum error, which can be the greatest outlier, will be compared to a set of thresholds (anomaly severities), that define each anomaly severity.

Figure 21 displays what a user would see in the logs of the Azure Function as the response to the requests.

```
New batch of sensor data sent...
Response: No anomalies detected.
-----
New batch of sensor data sent...
Response: No anomalies detected.
-----
New batch of sensor data sent...
Response: No anomalies detected.
-----
Response: Medium severity anomalies in the track!
```

Figure 21 - Anomaly detection and classification modules response

## 5.4 Final remarks

In this chapter, the author displayed a small prototype to detect and classify anomalies. Unfortunately, since there is no real data regarding railway tracks, it is not possible to elaborate on a more realistic simulation.

Regarding non-functional requirements, the usability requirements were met, since only authorized and authenticated users who are in the same Azure active directory tenant and have correct permissions can access the service bus. On the other hand, only the service bus and azure function are fault-tolerant, so the reliability requirements were partially met. Regarding performance, the algorithms used proved to be efficient during the tests. Logging was also implemented, so the supportability requirements were also met. Finally, regarding additional requirements, the anomaly detection and classification modules only allow HTTP requests and not HTTPS, so it was partially implemented.

## 6 Conclusion

In this chapter, the conclusions taken from the developed project will be addressed. It starts with a summary of the developed project, followed by the accomplished objectives, finishing with future work and improvements for the project.

### 6.1 Summary

This project aimed to detect and classify anomalies in a railway track and help build a predictive maintenance system, by training machine learning and/or deep learning models with a set of simulated data from sensors arranged in a train.

The first step to achieve the main objective of this project was to research similar approaches to this type of predictive maintenance system. This was the hardest part of the project since not many projects use multi-variant time-series datasets to train anomaly detection models. Furthermore, there was also the limitation of using semi-supervised and unsupervised learning, by feeding the model with only normal data.

Getting a sense of which are the most popular models through the elaboration of a state of the art on the area under analysis, the author was able to decide that the models he would try out to detect anomalies would be OC-SVM, CNN-AE, and LSTM-AE. For the autoencoders, different timesteps were also taken into account.

After training and testing the models, it was decided to use the LSTM-AE in the final prototype, given its recall.

Since an autoencoder was chosen, and the time frame to finish the project was short, it was decided that for the classification component of the project, different thresholds would be used to classify the anomaly.



In the end, the anomaly detection part of this project showed promising results, but the classification still requires more testing with different approaches.

## 6.2 Accomplished objectives

The following table summarizes the accomplished goals of the project:

Objective	Degree of achievement
Research anomaly detection techniques for predictive maintenance systems	Accomplished
Detect anomalies on the railway track	Accomplished
Research anomaly severity classification techniques for predictive maintenance systems	Accomplished
Classify anomalies on the railway track	Partially accomplished

## 6.3 Future work and improvements

The most important improvement and future work for this project is to improve the anomaly classification techniques. For instance, clustering techniques could be tested, where each cluster would group anomalies by severity.

Another future work would be to test online learning models since predictive maintenance systems benefit from it.

Testing the models with real scenarios would be another important achievement in the future.

Finally, having a better flow to read the sensor data, and a user interface to see the results, would also be an interesting evolution.

# Bibliography

- [1] “FEUP - Projeto/Contrato PS:POCI-01-0247-FEDER-046111.” [https://sigarra.up.pt/feup/pt/projectos\\_geral.mostra\\_projecto?P\\_ID=77312](https://sigarra.up.pt/feup/pt/projectos_geral.mostra_projecto?P_ID=77312) (accessed Dec. 21, 2022).
- [2] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. S. Alcalá, “A systematic literature review of machine learning methods applied to predictive maintenance,” *Comput Ind Eng*, vol. 137, p. 106024, Nov. 2019, doi: 10.1016/J.CIE.2019.106024.
- [3] M. Fernandes, J. M. Corchado, and G. Marreiros, “Machine learning techniques applied to mechanical fault diagnosis and fault prognosis in the context of real industrial manufacturing use-cases: a systematic literature review,” *Applied Intelligence*, vol. 52, no. 12, pp. 14246–14280, Sep. 2022, doi: 10.1007/S10489-022-03344-3/FIGURES/9.
- [4] J. Dalzochio *et al.*, “Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges,” *Comput Ind*, vol. 123, p. 103298, Dec. 2020, doi: 10.1016/J.COMPIND.2020.103298.
- [5] C. Vos, B. Eiteneuer, and O. Niggemann, “Incorporating Uncertainty into Unsupervised Machine Learning for Cyber-Physical Systems,” *Proceedings - 2020 IEEE Conference on Industrial Cyberphysical Systems, ICPS 2020*, pp. 475–480, Jun. 2020, doi: 10.1109/ICPS48405.2020.9274779.
- [6] “What is Unsupervised Learning? | IBM.” <https://www.ibm.com/cloud/learn/unsupervised-learning> (accessed Dec. 15, 2022).
- [7] J. E. van Engelen and H. H. Hoos, “A survey on semi-supervised learning,” *Mach Learn*, vol. 109, no. 2, pp. 373–440, Feb. 2020, doi: 10.1007/S10994-019-05855-6/FIGURES/5.
- [8] A. Lourenço, J. Meira, and G. Marreiros, “Online adaptive learning for out-of-round railway wheels detection,” *Proceedings of the ACM Symposium on Applied Computing*, pp. 418–421, 2023, doi: 10.1145/3555776.3577860.
- [9] J. Meira, B. Veloso, V. Bolón-Canedo, G. Marreiros, A. Alonso-Betanzos, and J. Gama, “Data-driven predictive maintenance framework for railway systems,” *Intelligent Data Analysis*, vol. 27, pp. 1087–1102, 2023, doi: 10.3233/IDA-226811.
- [10] J. Meira, C. Eiras-Franco, V. Bolón-Canedo, G. Marreiros, and A. Alonso-Betanzos, “Fast anomaly detection with locality-sensitive hashing and hyperparameter autotuning,” *Inf Sci (N Y)*, vol. 607, pp. 1245–1264, 2022, doi: 10.1016/j.ins.2022.06.035.

- [11] B. Krawczyk, M. Galar, M. Woźniak, H. Bustince, and F. Herrera, "Dynamic ensemble selection for multi-class classification with one-class classifiers," *Pattern Recognit*, vol. 83, pp. 34–51, Nov. 2018, doi: 10.1016/J.PATCOG.2018.05.015.
- [12] S. Alam, S. K. Sonbhadra, S. Agarwal, and P. Nagabhushan, "One-class support vector classifiers: A survey," *Knowl Based Syst*, vol. 196, p. 105754, May 2020, doi: 10.1016/J.KNOSYS.2020.105754.
- [13] J. Saari, D. Strömbergsson, J. Lundberg, and A. Thomson, "Detection and identification of windmill bearing faults using a one-class support vector machine (SVM)," *Measurement*, vol. 137, pp. 287–301, Apr. 2019, doi: 10.1016/J.MEASUREMENT.2019.01.020.
- [14] Z. Cheng, C. Zou, and J. Dong, "Outlier detection using isolation forest and local outlier," *Proceedings of the 2019 Research in Adaptive and Convergent Systems, RACS 2019*, pp. 161–168, Sep. 2019, doi: 10.1145/3338840.3355641.
- [15] J. Lesouple, C. Baudoin, M. Spigai, and J.-Y. Tourneret, "Generalized isolation forest for anomaly detection," *Pattern Recognit Lett*, vol. 149, pp. 109–119, Sep. 2021, doi: 10.1016/J.PATREC.2021.05.022.
- [16] F. Zhang, Z. Hong, T. Gao, and S. Yin, "A Fault Detection Method for Analog Circuits Based on the Wavelet Features and One-class KNN," *ICSMD 2021 - 2nd International Conference on Sensing, Measurement and Data Analytics in the Era of Artificial Intelligence*, 2021, doi: 10.1109/ICSMD53520.2021.9670762.
- [17] D. H. Hoang and H. D. Nguyen, "A PCA-based method for IoT network traffic anomaly detection," *International Conference on Advanced Communication Technology, ICACT*, vol. 2018-February, pp. 381–386, Mar. 2018, doi: 10.23919/ICACT.2018.8323766.
- [18] L. Ruff *et al.*, "Deep One-Class Classification." PMLR, pp. 4393–4402, Jul. 03, 2018. Accessed: Jan. 21, 2023. [Online]. Available: <https://proceedings.mlr.press/v80/ruff18a.html>
- [19] A. L. Alfeo, M. G. C. A. Cimino, G. Manco, E. Ritacco, and G. Vaglini, "Using an autoencoder in the design of an anomaly detector for smart manufacturing," *Pattern Recognit Lett*, vol. 136, pp. 272–278, Aug. 2020, doi: 10.1016/J.PATREC.2020.06.008.
- [20] E. H. Budiarto, A. Erna Permanasari, and S. Fauziati, "Unsupervised anomaly detection using K-Means, local outlier factor and one class SVM," *Proceedings - 2019 5th International Conference on Science and Technology, ICST 2019*, Jul. 2019, doi: 10.1109/ICST47872.2019.9166366.
- [21] T. Kansal, S. Bahuguna, V. Singh, and T. Choudhury, "Customer Segmentation using K-means Clustering," *Proceedings of the International Conference on Computational Techniques, Electronics and Mechanical Systems, CTEMS 2018*, pp. 135–139, Dec. 2018, doi: 10.1109/CTEMS.2018.8769171.
- [22] K. Khan, S. U. Rehman, K. Aziz, S. Fong, S. Sarasvady, and A. Vishwa, "DBSCAN: Past, present and future," *5th International Conference on the Applications of Digital Information and Web Technologies, ICADIWT 2014*, pp. 232–238, 2014, doi: 10.1109/ICADIWT.2014.6814687.
- [23] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, Oct. 2021, doi: 10.1016/J.NEUCOM.2021.04.112.

- [24] M. Gusenbauer and N. R. Haddaway, "Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources," *Res Synth Methods*, vol. 11, no. 2, pp. 181–217, Mar. 2020, doi: 10.1002/JRSM.1378.
- [25] F. Farbiz, Y. Miaolong, and Z. Yu, "A Cognitive Analytics based Approach for Machine Health Monitoring, Anomaly Detection, and Predictive Maintenance," *Proceedings of the 15th IEEE Conference on Industrial Electronics and Applications, ICIEA 2020*, pp. 1104–1109, Nov. 2020, doi: 10.1109/ICIEA48937.2020.9248409.
- [26] N. Amruthnath and T. Gupta, "A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance," *2018 5th International Conference on Industrial Engineering and Applications, ICIEA 2018*, pp. 355–361, Jun. 2018, doi: 10.1109/IEA.2018.8387124.
- [27] T. Hiruta, K. Maki, T. Kato, and Y. Umeda, "Unsupervised Learning Based Diagnosis Model for Anomaly Detection of Motor Bearing with Current Data," *Procedia CIRP*, vol. 98, pp. 336–341, Jan. 2021, doi: 10.1016/J.PROCIR.2021.01.113.
- [28] Y. Tan, C. Niu, H. Tian, L. Hou, and J. Zhang, "A one-class SVM based approach for condition-based maintenance of a naval propulsion plant with limited labeled data," *Ocean Engineering*, vol. 193, p. 106592, Dec. 2019, doi: 10.1016/J.OCEANENG.2019.106592.
- [29] M. Arias Chao, B. T. Adey, and O. Fink, "Implicit supervision for fault detection and segmentation of emerging fault types with Deep Variational Autoencoders," *Neurocomputing*, vol. 454, pp. 324–338, Sep. 2021, doi: 10.1016/J.NEUCOM.2021.04.122.
- [30] M. L. Hoffmann Souza, C. A. da Costa, G. de Oliveira Ramos, and R. da Rosa Righi, "A feature identification method to explain anomalies in condition monitoring," *Comput Ind*, vol. 133, p. 103528, Dec. 2021, doi: 10.1016/J.COMPIND.2021.103528.
- [31] K. Michalowska, S. Riemer-Sorensen, C. Sterud, and O. M. Hjellset, "Anomaly Detection with Unknown Anomalies: Application to Maritime Machinery," *IFAC-PapersOnLine*, vol. 54, no. 16, pp. 105–111, Jan. 2021, doi: 10.1016/J.IFACOL.2021.10.080.
- [32] C. M. A. Roelofs, M. A. Lutz, S. Faulstich, and S. Vogt, "Autoencoder-based anomaly root cause analysis for wind turbines," *Energy and AI*, vol. 4, p. 100065, Jun. 2021, doi: 10.1016/J.EGYAI.2021.100065.
- [33] Z. Li, Y. Sun, L. Yang, Z. Zhao, and X. Chen, "Unsupervised Machine Anomaly Detection Using Autoencoder and Temporal Convolutional Network," *IEEE Trans Instrum Meas*, vol. 71, 2022, doi: 10.1109/TIM.2022.3212547.
- [34] N. Kolokas, T. Vafeiadis, D. Ioannidis, and D. Tzovaras, "Anomaly Detection in Aluminium Production with Unsupervised Machine Learning Classifiers," *IEEE International Symposium on INnovations in Intelligent SysTems and Applications, INISTA 2019 - Proceedings*, Jul. 2019, doi: 10.1109/INISTA.2019.8778419.
- [35] C. Vos, B. Eiteneuer, and O. Niggemann, "Incorporating Uncertainty into Unsupervised Machine Learning for Cyber-Physical Systems," *Proceedings - 2020 IEEE Conference on Industrial*

*Cyberphysical Systems, ICPS 2020*, pp. 475–480, Jun. 2020, doi: 10.1109/ICPS48405.2020.9274779.

- [36] E. De Santis, F. Arno, A. Martino, and A. Rizzi, “A statistical framework for labeling unlabelled data: a case study on anomaly detection in pressurization systems for high-speed railway trains,” *Proceedings of the International Joint Conference on Neural Networks*, vol. 2022-July, 2022, doi: 10.1109/IJCNN55064.2022.9892880.
- [37] L. Prieto-Gonzalez, G. Tamm, and V. Stantchev, “Towards a software engineering approach for cloud and IoT services in healthcare,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9789, pp. 439–452, 2016, doi: 10.1007/978-3-319-42089-9\_31.
- [38] “Cloud Computing Services | Microsoft Azure.” <https://azure.microsoft.com/en-us> (accessed Jul. 22, 2023).
- [39] “MATLAB - MathWorks - MATLAB & Simulink.” <https://www.mathworks.com/products/matlab.html> (accessed Jan. 22, 2023).
- [40] “Python.org.” <https://www.python.org/> (accessed Jan. 22, 2023).
- [41] “Keras: Deep Learning for humans.” <https://keras.io/> (accessed Jul. 22, 2023).
- [42] “TensorFlow.” <https://www.tensorflow.org/> (accessed Jul. 22, 2023).
- [43] T. O’Malley *et al.*, “KerasTuner.” 2019.
- [44] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011, [Online]. Available: <http://jmlr.org/papers/v12/pedregosa11a.html>
- [45] “Apresentação\_FEUP\_ISEP\_LNEC\_24.”
- [46] A. Patle and D. S. Chouhan, “SVM kernel functions for classification,” *2013 International Conference on Advances in Technology and Engineering, ICATE 2013*, 2013, doi: 10.1109/ICADTE.2013.6524743.
- [47] “RBF SVM parameters — scikit-learn 1.3.0 documentation.” [https://scikit-learn.org/stable/auto\\_examples/svm/plot\\_rbf\\_parameters.html](https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html) (accessed Jul. 22, 2023).
- [48] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, “1D convolutional neural networks and applications: A survey,” *Mech Syst Signal Process*, vol. 151, p. 107398, 2021, doi: 10.1016/j.ymssp.2020.107398.
- [49] A. Baldominos, Y. Saez, and P. Isasi, “Evolutionary design of convolutional neural networks for human activity recognition in sensor-rich environments,” *Sensors (Switzerland)*, vol. 18, no. 4, 2018, doi: 10.3390/s18041288.
- [50] A. Senior, “Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling Has”.

- [51] G. Murtaza *et al.*, “Deep learning-based breast cancer classification through medical imaging modalities: state of the art and research challenges,” *Artif Intell Rev*, vol. 53, no. 3, pp. 1655–1720, Mar. 2020, doi: 10.1007/S10462-019-09716-5/TABLES/3.
- [52] D. K. Sharma, M. Chatterjee, G. Kaur, and S. Vavilala, “Deep learning applications for disease diagnosis,” *Deep Learning for Medical Applications with Unique Data*, pp. 31–51, Jan. 2022, doi: 10.1016/B978-0-12-824145-5.00005-8.
- [53] F. Demir, “Deep autoencoder-based automated brain tumor detection from MRI data,” *Artificial Intelligence-Based Brain-Computer Interface*, pp. 317–351, Jan. 2022, doi: 10.1016/B978-0-323-91197-9.00013-8.